VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra elektroniky

Inovace laboratorních úloh z předmětu ČMT2

Innovation of Laboratory Exercises from the DMT2

2020 Pavel Cyprich

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra elektroniky

Zadání bakalářské práce

Student: **Pavel Cyprich**

Studijní program: B0714A060012 Aplikovaná elektronika

Téma: Inovace laboratorních úloh z předmětu ČMT2
Innovation of Laboratory Exercises from the DMT2

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Zpracujte zadání a teoretický rozbor jednotlivých laboratorních úloh podle pokynů vedoucího práce.
2. Vypracujte vzorové protokoly k jednotlivým úlohám.
3. Vytvořte podpůrné materiály a návody pro vyučujícího.

Seznam doporučené odborné literatury:

VÁŇA, Vladimír. ARM pro začátečníky. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-246-6.
KADLEC, Václav. Učíme se programovat v jazyce C: základ pro programování v C++, C#, Javě, JavaScriptu, PHP a jiných jazycích. Praha: Computer Press, c2002. Programování. ISBN 80-7226-715-9.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Martin Sobek, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

doc. Ing. Petr Palacký, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *7. května 2020* .....

podpis studenta

Poděkování

Tímto bych rád poděkoval panu Ing. Martinu Sobkovi, Ph.D. za odborné vedení mé bakalářské práce, za cenné rady a čas, který mi věnoval během osobních konzultací.

Také musím poděkovat rodině, bez které by práce nemohla vzniknout.

Abstrakt

Cílem bakalářské práce bylo vytvoření laboratorních úloh do předmětu ČMT2, dále poskytnout studentům podpůrný materiál a seznámit je tak zároveň se základními principy činnosti, konfigurací a také funkcemi základních periferních obvodů a to prostřednictvím napsaného programového kódu k vybrané aplikaci s kompletním návodem pro její vypracování. Hlavní náplní práce je tedy dostatečně studenta seznámit o používaných komunikačních protokolech a nejčastěji využívaných periferních obvodech v dnešní mikroprocesorové technice.

V úvodní části je představen vývojový nástroj Processor Expert. Práce se z velké části zabývá rozborem vytvořených laboratorních úloh, jehož součástí je představení úlohy spolu s metodou vypracování. Nachází se zde vysvětlení, na co je konkrétní aplikace zaměřena, ke kterým výsledkům má student dospět a co si po jejím vypracování otestuje. Nechybí zde předvedení způsobu použití vybraných funkcí knihoven. Nakonec jsou zmíněny požadavky na vytvoření protokolů spolu s konfigurací nejčastěji využívaných knihovních komponent.

Klíčová slova

Mikrokontrolér, Sběrnice, Processor Expert, Komponenta, Komunikace, Čítač, Časovač

Abstract

The aim of the bachelor thesis was to create laboratory tasks for the subject CMT2 and to provide students support material and acquaint them with the basic principles of operation, configuration and functions of basic peripherals through the written program code for selected application with complete instructions for its elaboration. The main goal of the thesis is to acquaint the student sufficiently with the communication protocols and the most commonly used peripheral circuits in modern electronics.

The introductory part introduces the development tool Processor Expert. The work is largely concerned with the analysis of laboratory exercises, which includes the introduction of the task together with the method of elaboration. There is an explanation of what a particular application is aimed at, what results the student should achieve and what the student will test after its elaboration. There is also a demonstration of the use of selected library functions. Finally, the requirements for creating protocols together with the configuration of the most frequently used library components are mentioned.

Keywords

Microcontroller, Bus, Processor Expert, Component, Communication, Counter, Timer

Obsah

Seznam použitých symbolů

Symbol	Jednotky	Význam symbolu
BaudRate	bit/s	Přenosová rychlost
f	Hz	Frekvence
R	Ω	Odpor
T	s	Perioda
U	V	Napětí

Seznam použitých zkratek

Zkratka	Význam
ADC	Analog to Digital Converter
ASCII	American Standard Code for Information Interchange
COM	Communication port
CPHA	Clock Phase - fáze hodinového signálu
CPOL	Clock Polarity - polarita hodinového signálu
CPU	Procesor
ČMT	Číslicová a mikroprocesorová technika
D/A, DAC	Digital to Analog Converter - digitálně analogový převodník
FLL	Frequency-locked look - fázový závěs
FTM	Flex Timer Module - speciální časovací modul
GPIO	General Purpose Input Output
HEX	Hexadecimal - zkratka užívaná pro zápis hexadecimálních čísel
I/O	Vstupně výstupní pin
I2C	Internal Integrated Circuit Bus - sériová datová sběrnice
IRC	Internal Reference Clock - vnitřní zdroj hodinového signálu
IRC48M	Internal Reference Clock 48 MHz
LCD	Liquid Crystal Display
LPTMR	Low Power Timer

LSBit	Nejméně významový bit
MCG	Multipurpose Clock Generator- blok pro výběr zdroje hodinového signálu
MCU	Microcontroller Unit - mikropočítač
MISO, MOSI	Master In Slave Out pin, Master Out Slave In pin
MOSFET	Metal Oxide Semiconductor Field Effect Transistor - polem řízený tranzistor
MSBit	Nejvíce významový bit
NVIC	Nested Vectored Interrupt Controller - řadič přerušení
OpenSDA	Open-standard Serial and Debug Adapter
PC	Počítač
PDB	Programmable Delay Block
PE	Processor Expert - nástroj pro vývoj embedded aplikací
PEE	PLL Engaged External
PHY	Physical Layer
PIT	Programmable Interval Timer
PLL	Phase-locked loop
PWM	Pulse Width Modulation - pulzní šířková modulace
RAM	Random Access Memory - paměť pro krátkodobé uchování dat
RS422, RS485	Standardy sériové komunikace
RTC	Real Time Clock - hodiny reálného času
RxD	Pin pro příjem dat na UART sběrnici
SCK, SCL	Serial Clock Line - označení pinu pro hodinový signál
SDA	Serial Data Line - sériová datová linka
SIM	System Integration Module - modul pro správu časování vnitřních sběrnic mikrokontroléru
SPI	Serial Peripheral Interface
SS	Slave Select - pin pro výběr připojeného zařízení na sběrnici
TxD	Pin pro vysílání dat na UART sběrnici
UART	Universal Asynchronous Receiver and Transmitter
USART	Universal Synchronous / Asynchronous Receiver and Transmitter
USB	Universal Serial Bus

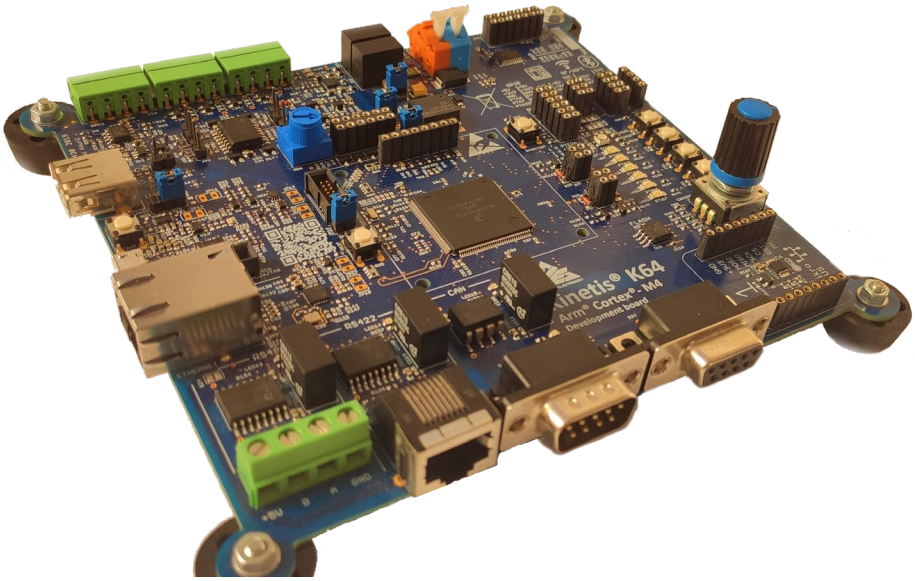
Seznam ilustrací a seznam tabulek

Úvod

Tato práce je zaměřena na vytvoření výukových materiálů a laboratorních úloh do studijního předmětu Číslicová a mikroprocesorová technika 2 (ČMT2). Předmět je určen pro studenty magisterských studijních programů, vyskytuje se ve studijních plánech oborů aplikovaná elektronika, elektronika. Předmět se zaměřuje na poznatky z číslicové techniky a mikroprocesorové techniky. Náplň předmětu se opírá o poznatky z teorie elektronických obvodů, elektroniky a analogové techniky.

Během výuky je potřeba studenta seznámit s principy konfigurace a také funkce a práce základních periferních obvodů, jakými jsou moduly GPIO, čítačů, časovačů, UART, SPI, I2C využívaných v 32bitových mikropočítačových systémech. Ve své práci se zaměřuji především na vysvětlení principů funkce a využití nejpoužívanějších komunikačních protokolů. K tomu využívám napsané laboratorní úlohy, které se zaměřuje na konkrétní problematiku. Student je veden k používání vývojového nástroje Processor Expert, od společnosti NXP, určeného pro tvorbu vestavěných aplikací. Díky tomu je student schopen flexibilně reagovat na vzniklé problémy či chyby a sám najít cestu vedoucí ke správnému řešení.

K ověření funkčnosti napsaného kódu slouží bohatě vybavená vývojová deska s mikrokontrolerem MK64FN1M0VLQ12, která poskytuje velké množství vyvedených konektorů a rozhraní. Tyto prostředky jsou využívány pro připojení externích zařízení, za účelem, ať už se jedná o komunikaci s PC pomocí microUSB kabelu s komunikací prostřednictvím OpenSDA rozhraní, připojení displeje Nokia 3310, na kterém se studenti seznámí s principy komunikace po SPI, a také s jednoduchým zobrazováním textů a obrazců. Na vývojové desce také najdeme sadu led nebo integrovaný obvod teplotního senzoru, pomocí kterého se student seznámí s I2C sběrnicí a vyzkouší si tak její praktické využití pro měření teploty.



Obr. 1: *Vývojová deska s mikrokontrolérem MK64FN1M0VLQ12*

1 Processor Expert

1.1 Základní informace a využití Processor Expert

Processor Expert, dále jen PE, je vývojový nástroj vyvinut společností Unis v roce 1994 v České republice a v roce 2008 získán společností NXP. Zpočátku byl využíván ve vývojovém prostředí CodeWarrior, postupně však byl zahrnut i do ostatních prostředí společnosti NXP, jakými jsou Eclipse a Kinetis Design Studio. [7]

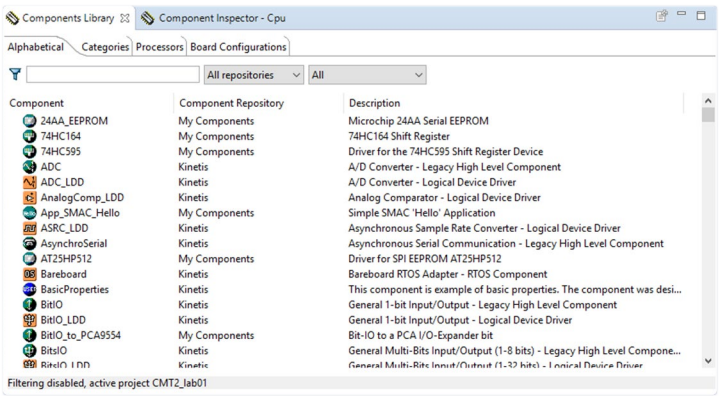
Jedná se tedy o doplněk vývojového prostředí usnadňující vývoj vestavěných aplikací používaných mikrokontrolérů společnosti NXP. PE obsahuje velké množství komponent, tedy knihoven a grafických nástrojů pro nastavení a konfigurace parametrů jednotlivých periferií mikrokontrolérů (ADC, DAC, UART, …) a externích zařízení, jakými mohou být např. externí paměti, různé druhy displejů, senzory, snímače a jiných, v grafické podobě. Tyto komponenty má možnost uživatel také vytvářet a nespoléhat se tak zcela na již předpřipravené nástroje.

Umožňuje nám jednoduše řídit a nakonec také měnit nastavené parametry kdykoliv v průběhu psaní našeho programu. PE informuje vývojáře o možnostech použitých pinů, periférií a hodin tak, aby byl vývoj programu co možná nejefektivnější. Uživatel nástroje se již nemusí v tak velké míře spoléhat na referenční manuál výrobce, avšak stále se neobejde bez znalosti používaného MCU a znalostí používaných nástrojů. [7] [8]

1.2 Knihovna komponent

Pro vyhledávání komponent nám slouží knihovna *Components Library*. Nacházejí se zde „bloky“ komponent označeny příslušnou ikonou. Dvojklikem na ikonu vybrané komponenty ji přidáme do našeho projektu. Zvolené komponenty jsou tak následně importovány do složky *Components*. V záložce „*Component Inspector*“ je poté možné zvolené bloky nastavit.

Bloky komponent můžeme rozdělit na celkem 3 úrovně. Komponenty vysoké úrovně, střední a nízké úrovně. Komponenty vysoké úrovně využívají, ve většině případů, sub-komponenty nižších úrovní a patří mezi nejjednodušeji používané, jelikož nám velkou část nastavení periférii provádí již za nás a také použití knihovních funkcí je, co se týče psaní programu, mnohem přehlednější, efektivnější a také intuitivnější.



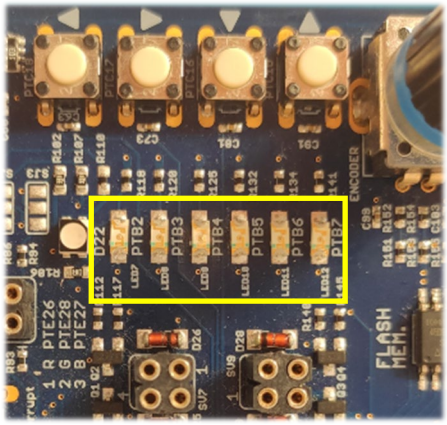
Obr. 2: *Knihovna komponent nástroje Processor Expert*

2 Rozbor laboratorních úloh

2.1 Generování obdélníkových průběhů

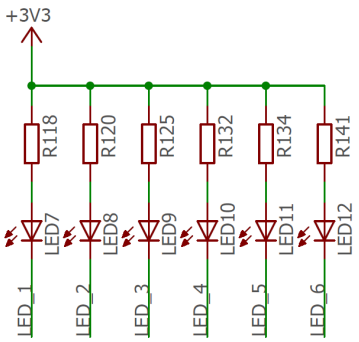
2.1.1 Představení laboratorní úlohy

Úloha je zaměřena na obeznámení studenta se základní koncepcí vytváření aplikačního softwaru využitím vývojového programu Kinetis Design Studio od NXP s rozšířením o PE. Úkolem studenta je vytvořit obdélníkový průběh využitím časového zpoždění pomocí CPU a využitím generovaného přerušení časovačem. Student je provázen postupem pro úspěšné založení programu a jeho tvorbu, nastavením časování vnitřních sběrnic MCU pro dosažení maximálního výpočetního výkonu CPU a připojených periférií. Student se také dozví o jednoduchých principech využití čítačů a časovačů, které jsou nedílnou součástí každého MCU. Seznámí se s nastavením a použitím vybraných komponent pro vytvoření obdélníkového průběhu generovaného na výstupních pinech. K tomu bude využívat led rozmístěných na desce podle obr. 3.



Obr. 3: *Rozložení led s popisem pinů na desce*

Vygenerované obdélníkové průběhy v závěru zachytí osciloskopem na výstupních pinech, ke kterým jsou led připojeny se společnými anodami k napájecímu napětí.



LED_1	PTB2
LED_2	PTB3
LED_3	PTB4
LED_4	PTB5
LED_5	PTB6
LED_6	PTB7

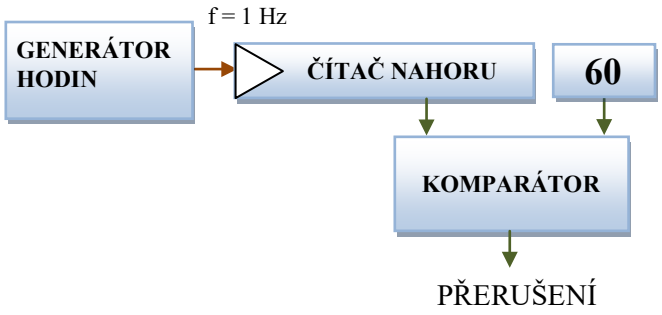
Obr. 4: *Schéma zapojení led s popisem vyvedení pinů mikrokontroléru na desce*

2.1.2 Čítače a časovače

Čítače a časovače patři neodmyslitelně k povinné výbavě každého dnes již vyráběného mikrokontroléru. Ve skutečnosti se jedná o jeden a tentýž periferní obvod neboli modul, skládající se z čítače impulzů a určitého počtu registrů pro vlastní nastavení (inicializaci), kontrolu stavu a řízení. Tyto obvody jsou vybaveny jedním nebo většinou i více kanály. Pokud čítač čítá impulzy z vnitřního obvodu oscilátoru mikrokontroléru, nazýváme celý modul jako časovač, naopak čítá-li impulzy z vnějšího zdroje přiváděného na vstupní pin (kanál), nazýváme jej již jako čítač. Princip funkce čítačů a časovačů je stejný u všech výrobců MCU. [1]

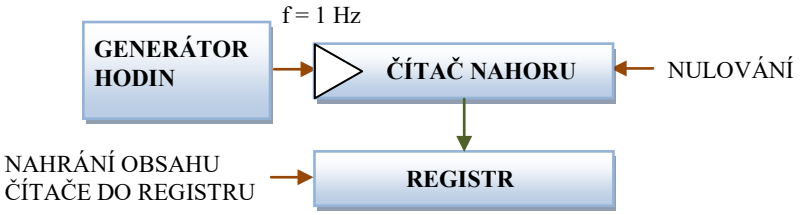
Využití časovače

Časovač využíváme nejčastěji pro generování časového zpoždění. Vnitřní čítač modulu se inkrementuje vnitřním hodinovým signálem, jehož frekvence je nastavena pomocí děliček frekvence z referenčního kmitočtu (1 kHz). Vnitřním komparátorem se neustále porovnává obsah čítače s přednastavenou hodnotou v registru (60). V okamžiku, dojde-li ke shodě, pak komparátor indikuje tuto rovnost např. nastavením určitého bitu stavového registru nebo také vygenerováním přerušení, pokud tuto možnost programátor sám v počáteční inicializaci povolí. [1]



Obr. 5: *Princip funkce časovače pro generování časového zpoždění (podle* [1]*)*

Další možností, jak časovač využít, je zjištění doby mezi dvěma po sobě jdoucími událostmi (zmíněná metoda se využívá např. k měření vzdálenosti ultrazvukovým senzorem nebo pro vyhodnocování rychlosti ze snímačů otáček). Pro vysvětlení nám může pomoci následující obrázek. Čítač modulu se opět inkrementuje vnitřním hodinovým signálem. Dojde-li k indikaci vnější událostí na vstupním pinu, tedy kanálu časovacího modulu, obsah čítače se okamžitě vynuluje. Obsah čítače začne narůstat od nulové počáteční hodnoty. Zaznamená-li se opět vnější událost, nahraje se celý obsah čítače do registru. Hodnota v registru pak udává délku trvání mezi dvěma událostmi. [1]



Obr. 6: *Princip funkce časovače pro zjištění doby mezi dvěma událostmi (podle* [1]*)*

Využití čítače

Čítač se využívá pro zjištění počtu vnějších událostí přivedených na vstupní pin. Generátorem vnějšího signálu může být senzor nebo i jednoduché tlačítko. Princip je jednoduchý, dojde- li ke zmáčknutí tlačítka, obsah čítače se inkrementuje o jedna. Nárůst obsahu čítače probíhá až do jeho vynulování nebo přetečení. Tak můžeme v každém okamžiku obsah vnitřního čítače číst, jelikož výrobce vybavuje časovací moduly speciálními registry, do kterých se ukládá aktuální obsah čítačů, pak je možné získaná data programově zpracovat. [1]

Obr. 7: *Princip funkce čítače (podle* [1]*)*

Možnosti generování obdélníkového průběhu

Pro generování časového zpoždění je možné využít dvou metod, u každé z nich budou představeny výhody a nevýhody jejich použití:

- **Využitím CPU k vykonávání instrukcí, jejichž délku vykonání známe**

  Výhody
  - jednodušší použití v programovém kódu

  Nevýhody
  - nelze přesně nastavit požadované časové zpoždění
  - velmi neefektivní využití výpočetního výkonu CPU = během zpožďovací rutiny je CPU zahlceno, a tak jej nemůžeme využít pro užitečnějších úkoly
  - mnohem vyšší spotřeba elektrické energie = absence využití v nízko příkonových aplikací

- **Pomocí přerušení časovače**

  Výhody
  - nezávislý chod časovače a CPU = časovač po nastaveném čase vyvolá přerušení, následně si vyžádá pozornost pro vyřízení obsluhy přerušení pomocí NVIC, mezi tím je tedy možné výpočetní výkon využít pro užitečnější úkoly
  - možnost nastavit velmi přesné načasování
  - využití speciálních funkcí časovačů jako je režim snížené spotřeby (LPTMR modul)
  - nastavení priority přerušení

  Nevýhody
  - je potřeba vědět, jak časovač správně inicializovat, to se velmi často neobejde bez nastudování potřebných informací v referenčním manuálu výrobce MCU
  - časování závislé na zdroji hodinového signálu

Jak je na první pohled patrné, výhody použití časovače převažují nad jeho nevýhodami, v případě využití CPU pro časové zpoždění je tomu přesně naopak.

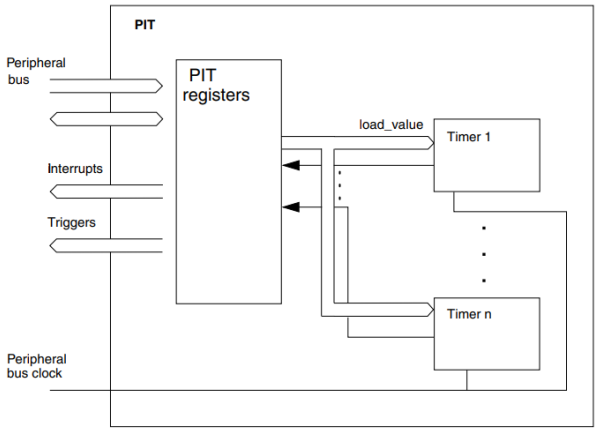
2.1.3 Periférie pro časování

Pro vytvoření periodického přerušení máme možnost použít *FlexTimer (*FTM*)*, Periodic *Interrupt Timer (*PIT*)*, *Programmable Delay Block (*PDB*)* nebo *Low Power Timer (*LPTMR*)*. Mikrokontrolér je rozšířen také o další časovací moduly, které nabízejí navíc speciální funkce.

PIT časovací modul [9] [2]

Jedná se o modul sdružující 32bit. časovače, které je možné, pomocí registrů modulu, nakonfigurovat pro periodické vyvolání přerušení. Přerušení každého z časovačů PIT modulu je vyhodnocováno zvlášť, proto je možné používat všechny časovače s rozdílným časem přerušení. Přerušení jsou maskovatelná, a tak jim můžeme přiřadit prioritu přerušení dle našich požadavků.

PIT modul pracuje velmi jednoduše. Po inicializaci se obsah každého zvoleného časovače přednastaví na hodnotu danou obsahem v registru PIT_LDVALn. V tom okamžiku se začíná obsah čítače dekrementovat s danou frekvencí z *bus clock* od zmíněné nastavené hodnoty. Po dosažení nulové hodnoty se vyvolá požadavek přerušení a nastaví se bit TIF registru PIT_TFLGn příslušného modulu na 1. Po vykonání obsluhy přerušení se čítač opět přenastaví na hodotu registru PIT_LDVALn, aby se tak mohl proces přerušení opakovat. Obsah časovače je možné v každém okamžiku monitorovat čtením registru PIT_CVALn.



Obr. 8: *Zjednodušené blokové schéma PIT časovacího modulu* [2]

LPTMR časovací modul

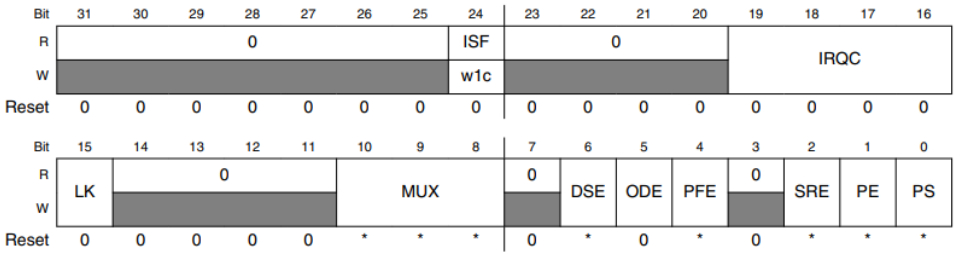
LPTMR je modul pracující s 16bit čítačem využívající se pro nízko příkonové aplikace, které nevyžadují rychlý chod (měření teploty okolí, vlhkosti vzduchu atd.). Své uplatnění nachází v *low-power* módech. K těmto účelům nabízí režimy nízké spotřeby a možnost dosažení velkého časového zpoždění. [2]

2.1.4 Funkce I/O pinů a jejich konfigurace

MCU je vybaven pěti paralelními porty, tedy moduly s paralelním uspořádáním I/O pinů. Každý port může být vybaven maximálně 32 piny. Tento počet můžeme porovnat se starším 8bitovým mikrokontrolérem 89s52, kde každý paralelní port je vybaven pouze osmi I/O piny. Většina pinů poskytuje výběr z mnoha alternativních funkcí. To umožňuje programátorovi stanovit, který pin bude zastávat požadovanou úlohu a vybrat také vhodné umístění. Najdeme zde také GPIO, tedy funkci pro základní použití pinu, kterým je zápis nebo čtení logického stavu.

Konfigurační registr PCR [1] [2]

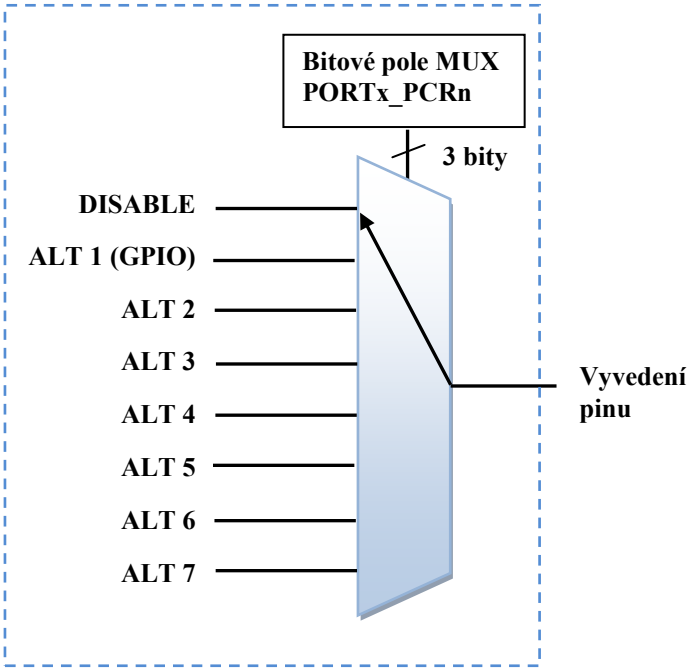
Výběr příslušné alternativní funkce pinu je umožněno speciálním funkčním registrem PCR. Ke každému pinu je možné přistupovat zvlášť samostatným funkčním registrem a zvolit, jinými slovy, správně nakonfigurovat, jeho požadovanou funkci.



Obr. 9: *Obsah konfiguračního registru PCR samostatného pinu* [2]

Výběr alternativní funkce je poskytnut multiplexováním. Toto řešení také nabízí úsporu počtu pinů a tedy redukci velikosti samotného MCU. Zmíněná konfigurace použitých pinů se provádí ihned po spuštění programu před samotným během aplikace, tedy během inicializace. Multiplexování se provádí vnitřním multiplexerem. V tom případě je tedy možné zvolit pouze jednu z nabízených funkcí, kterou pin ve stejném čase zastává.

Jeden pin je tak možné využít, tedy pokud danou funkci skutečně podporuje, jako GPIO, pin pro datový či hodinový synchronizační signál, pro vstupní a výstupní funkci modulů čítačů a časovačů, pro komunikační sběrnice nebo jako analogový I/O pin. I/O piny nabízejí alternativní funkce podle výbavy zvoleného MCU.



Obr. 10: *Výběr alternativní funkce pinu pomocí multiplexování (podle* [1, 2] *)*

Registrem PCR se navíc provádí volba zařazení vnitřního Pull-up nebo Pull-down odporu, povolení přerušení na pinu a jeho detekce, zařazení pasivního vstupního filtru a další. Před použitím pinu je nutné umožnit připojení zdroje hodinového signálu. Jinak jsou všechny piny automatický odpojeny a to za účelem snížení celkové spotřeby MCU.

Dále je každý paralelní port vybaven registrem pro nastavení směru pinu (PDDR) neboli pro nastavení vstupní či výstupní funkce. Pro ovládání pinů s nastavením funkce GPIO je po inicializaci uživateli poskytnuta široká škála registrů pro čtení logického stavu ze vstupního pinu (PDIR) či pro zápis logického stavu na výstupní pin (PDOR, PSOR, PCOR nebo PTOR).

2.1.5 Způsob vypracování a ověření výsledků

Vypracování

Aplikační software je vypracováván pomocí využití komponent LED, Wait, TimerInt a v neposlední řadě komponentou časování. Je důležité, aby se student jmenované komponenty naučil správně používat, jelikož ho budou doprovázet téměř každou laboratorní úlohou. Před používáním jakékoliv komponenty je potřeba vybrat a nastavit zdroj časování všech periférií mikrokontroléru a procesoru. Student je proveden konfigurací časového obvodu pro dosažení maximálního výpočetního výkonu, celý proces je prováděn za pomocí blokového schématu časovacího obvodu. Komponenta LED nabízí jednoduchou inicializaci zvoleného pinu jednotlivého GPIO modulu pro výstupní funkci ovládání led, zapojené na anodu nebo katodu na straně pinu. Komponenta také umožňuje základní generování PWM signálu pro ovládání jasu led nebo řízení počátečního stavu. Její součástí je také referenční komponenta BitIO pro nastavení požadovaného pinu GPIO modulu s nastavením směru a zvolení možnosti optimalizace. Pro vytvoření časového zpoždění pomocí CPU, slouží komponenta Wait. Její použití je velmi intuitivní a po importování z knihovny komponent nevyžaduje dodatečná nastavení. Komponenta TimerInt je zaměřena na ovládání zvoleného časovacího modulu a generování časového zpoždění pomocí přerušení časovače. Kromě volby zdroje přerušení umožňuje nastavit požadovaný čas přerušení, nastavit jeho prioritu, režim spotřeby a způsob inicializace.

Pro počáteční seznámení s programováním mikrokontrolérů ARM je na naší desce s mikrokontrolérem připojeno 6 led v zapojení se společnou anodou. Tyto led je možné ovládat přivedením log. 1 (led zhasnuta) a log. 0 (led rozsvícena) na příslušný pin portu B. Tak si student ověří správnost napsaného kódu a doloží jej nakonec snímkem z obrazovky osciloskopu.

Používané funkce komponent

LED **(** LED **)**

- *LED_Neg()* = negace aktuálního stavu na led
- *LED_On()* = rozsvícení led
- *LED_Off()* = zhasnutí led

Wait **(** Wait **)**

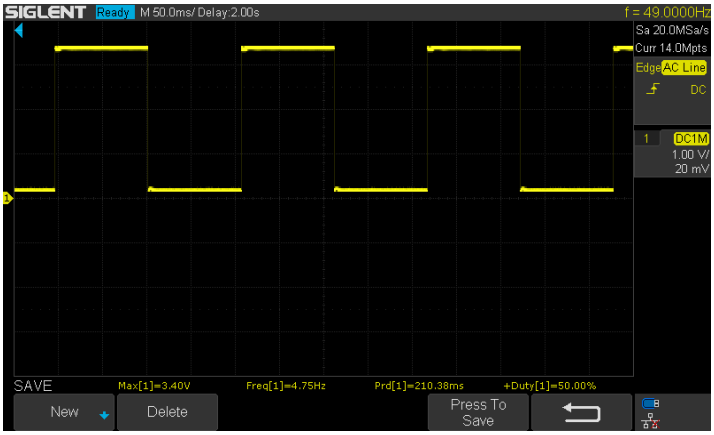
- *WAIT_Waitms* = vytvoření požadovaného časového zpoždění zadávaného v milisekundách

TimerInt **(** TimerInt **)**

- *TI_Enable()* = programové spuštění časovače s již nastaveným časem přerušení
- *TI_EnableEvent()* = povolení přerušení
- *TI_DisableEvent()* = zakázání přerušení
- *TI_OnInterrupt()* = událost vyvolaná přerušením časovače

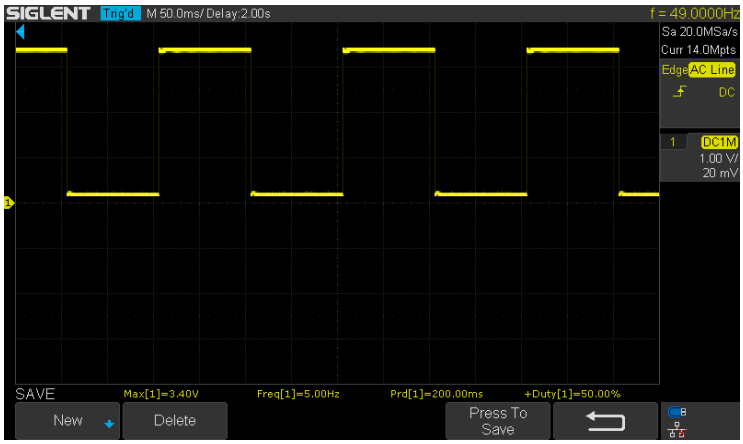
Ověření výsledků

Na obr. 11 je zachycen vygenerovaný obdélníkový průběh na připojené led s časovým zpožděním vytvořeným pomocí CPU. Použitím funkce LED_On() přímo přivádíme log. 0 z výstupního pinu pro rozsvícení led a naopak funkcí LED1_Off() log. 1 pro zhasnutí led. Processor Expert následně vygeneruje programový kód pro inicializaci vybraných pinů dle naší zvolené konfigurace, ten se zpracuje automaticky hned po spuštění programu nebo po resetu. Použitím WAIT_Waitms() bylo požadováno časové zpoždění 100 ms pro dosažení periody generovaného obdélníkového signálu 200 ms. Snímek osciloskopu ovšem potvrzuje hlavní nevýhodu použití této metody zpoždění, kterou je nepřesnost nastavení zpožďovací rutiny, jelikož jsme dospěli k celkové chybě nastavené periody 10,38 ms.



Obr. 11: *Generování obdélníkového signálu pomocí CPU*

Dalším úkolem je vytvoření obdélníkového signálu pomocí přerušení časovače. Pro náš úkol slouží časovač PIT_LDVAL0 s nastavením přerušení 100 ms. Po vyvolání přerušení se zpracovává událost TI1_OnInterrupt() naší komponenty, do které je vložen podprogram pro negaci připojené led, tedy funkce LED_Neg(). Jak je zobrazeno na obr. 12, ve výsledku došlo k vytvoření přesného obdélníkového signálu s periodou 200 ms.



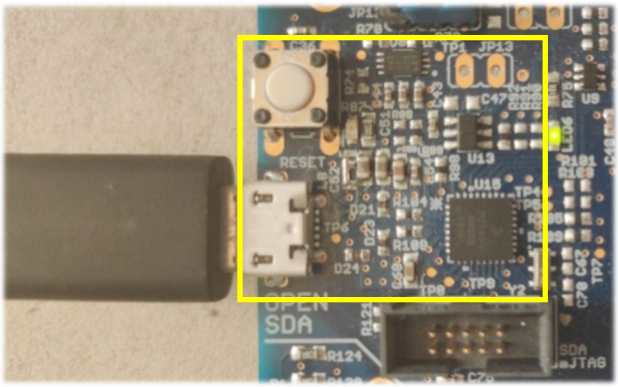
Obr. 12: *Generování obdélníkového signálu pomocí přerušení časovače*

Porovná-li student na závěr oba způsoby generovaných signálů, ověří si, že použitím časovače, jako zdroje časového zpoždění, dosahujeme přesnějších výsledků. Použitím časovače také nezatěžujeme výpočetní výkon CPU a můžeme jej tedy využít pro jiné výpočetní úlohy.

2.2 Komunikace mikropočítače s uživatelem pomocí sériové linky

2.2.1 Představení laboratorní úlohy

Následující úloha si klade za cíl seznámit studenta s principy komunikace po sériové lince prostřednictvím OpenSDA rozhraní, provést ho možností realizace programového kódu a možností konfigurace pinů s využitím vývojového nástroje PE a komponenty, pomocí níž lze nastavit parametry přenosu, řídit přenos dat po sériové lince a provádět formátování. Student má za úkol nejprve vypsat textový řetězec na sériový terminál. Dále bude muset najít způsob, jak vypsat textově číselnou hodnotu, jak načítat text obsahující číslo a nakonec vytvořit jednoduchou kalkulačku, jejímž úkolem je vykonávání základních matematických operací. Pro ověření svých výsledků používá program Terminal, který mu umožní sledovat a zapisovat data přes sériovou linku. Rovněž je zde kladen důraz na pochopení napsaného kódu, které student musí doložit jak popisem funkce programového kódu, tak napsaným závěrem, kdy se má vyjádřit k dosaženým výsledkům.



Obr. 13: *Rozhraní OpenSDA na desce*

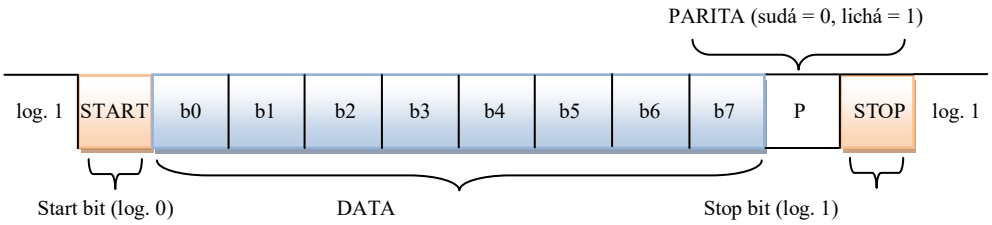
2.2.2 USART

Jedná se o univerzální synchronní nebo asynchronní komunikaci pro obousměrný přenos, tedy komunikaci probíhající v plně duplexním režimu. Přenos dat probíhá po dvou vodičích (RxD a TxD). Tato sériová komunikace využívá dvě metody, asynchronní (UART), někdy označována jako SCI, která je plně duplexní a synchronní (SPI) schopna také pracovat v plně duplexním režimu.

UART

Sběrnice využívána již od 60. let. Vyznačuje se jednoduchostí použití. Přenos dat se uskutečňuje asynchronně po sériové sběrnici maximální přenosovou rychlostí (BaudRate) 1 Mbit/s.

Vysílání dat se uskutečňuje z výstupního pinu označovaného jako TxD, přijímaní dat se provádí pinem RxD. Uživatel má většinou možnost výběru počtu přenášených bitů datového slova (obvykle 7bitové, 8bitové nebo 9bitové, ale můžeme se setkat i s větším počtem bitů datového slova). K vysílanému datovému slovu se připojuje Start bit, Stop bit a Paritní bit, aby se tak rozlišil začátek a konec příjmu dat pro zařízení, kterému se data posílají. [1]



Obr. 14: *Zorganizovaná data pro přenos po SCI*

Průběh komunikace [1] [10]

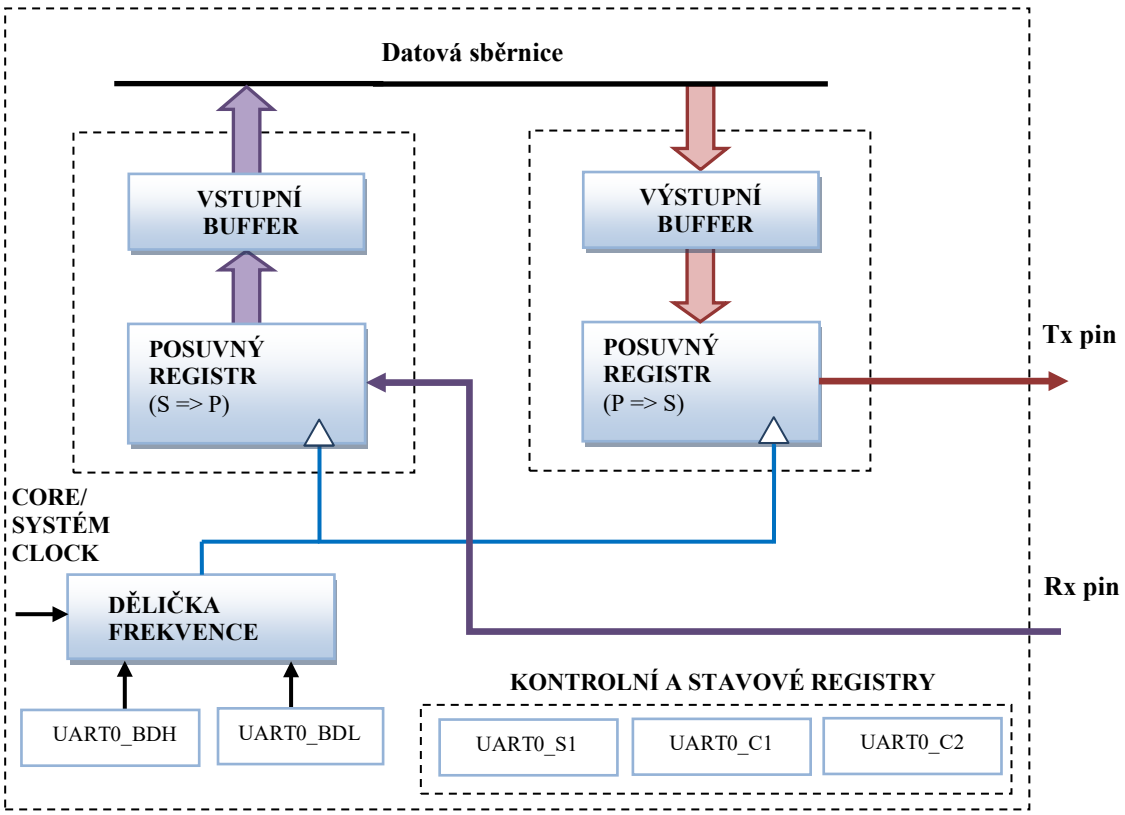
Klidová úroveň komunikačního signálu je log. 1. Vysílaná data jsou získávána z datové sběrnice MCU. Data jsou tedy převzata jako paralelní, pro jejich následný přenos je potřeba tato data převést na sériová. Tento převod se uskutečňuje v posuvných registrech našeho UART modulu. Během převodu se k vysílaným datům připojuje Start bit, Stop bit nebo také paritní bit a nakonec jsou data vyslána z výstupního TxD pinu, tehdy dochází k vysílání dat bit po bitu zvolenou přenosovou rychlostí. Nejprve se vysílá Start bit, který je reprezentován změnou z klidového stavu (log. 1) na log. 0 po dobu jednoho hodinového impulsu, poté následuje nejméně významový bit datového slova až do bitu nejvýznamnějšího, následuje volitelný parity bit a konec vysílaných dat je indikováno Stop bitem. Stop bit je reprezentován úrovní log. 1 po dobu jednoho nebo i dvou hodinových impulsů, Délka trvání tohoto konečného bitu záleží na možnostech zařízení, s kterým pomocí MCU komunikujeme, je totiž možné, že zařízení bude o něco pomalejší a bude tento bit navíc potřebovat k zorganizování námi přijatých dat v paměti.

Druhé zařízení pak data přijímá následovně. Detekuje-li Start bit, na svém přijímacím pinu RxD, začnou se následující bity číst s danou frekvencí přenosu vyjádřenou v bitech za sekundu (bit/s), Jelikož je přenos asynchronní, pak oběma zařízením účastnícím se komunikace, nastavujeme stejnou přenosovou rychlost, jinak by došlo ke ztrátě jednotlivých bitů dat během přenosu nebo v opačném případě přijetí chybné informace. Rozdíl přenosových frekvencí může činit pouze okolo 10 %. Na přijímací straně (na pinu RxD) dochází po přijetí datového slova k převedení sériových dat zpátky na paralelní a to opět pomocí posuvného registru. Během převodu nastává odstranění Start, Stop a paritního bitu. Paralelní data jsou nakonec poslána na datovou sběrnici přijímaného zařízení.

UART moduly společnosti NXP [1] [2]

Pro řízení, ovládání a tedy celou správu přenosu dat přes UART sběrnici jsou tyto moduly vybaveny sadou registrů. Jedná se o konfigurační registry (UARTx_C1, UARTx_C2, UARTx_BDH, UARTx_BDL). Ty nám umožňují nastavení parametrů přenosu, tedy určení přenosové rychlosti, dále pak výběr operačních módů, nastavení organizace dat (paritní bit, velikost datového slova, Stop bit) nebo také povolení přerušení. Dále jsou to stavové registry (UARTx_S1, UARTx_S2), které nás informují o stavu přenosu. Využívají se pro vyšetření aktuálního stavu posuvných registrů, tedy zda je daný registr prázdný a tedy např. připraven pro vysílání nových dat. Této indikace můžeme využít zvláště, pokud nehodláme používat přerušení, můžeme se tak mimo jiné dotazovat na zaslání posledního přenášeného bitu. V neposlední řadě to jsou registry, do nichž zapisujeme vysílaná data (UARTx_Tx) nebo naopak data čteme (UARTx_Rx). Výrobce MCU často také poskytuje sadu speciálních registrů.

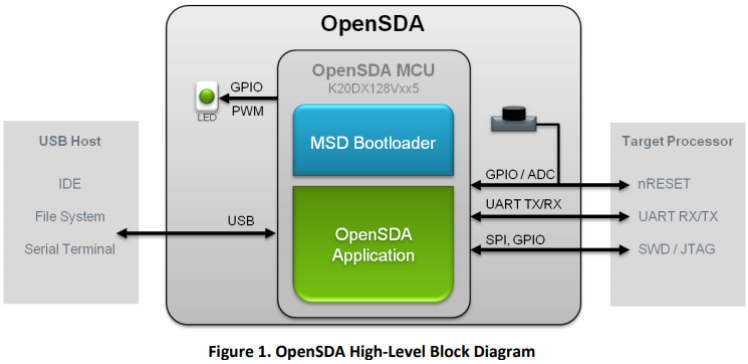
Tyto stavové, konfigurační, a také datové registry můžeme najít u všech UART periférií každého výrobce MCU, avšak často pod jiným označení nebo poněkud s odlišným způsobem správy komunikace.



Obr. 15: *Zjednodušené blokové schéma UART0 modulu (podle* [1]*)*

SCI rozhraní mikrokontroléru

Pro připojení mikrokontroléru k PC využíváme rozhraní OpenSDA *(open-standard serial and debug adapter*). *Rozhraní obsahuje bootloader (zavaděč), který poskytuje rychlý a snadný mechanismus pro načítání různých OpenSDA aplikací, jakými jsou např. ladící rozhraní (debug), flash programátory, převodníky mezi USB a sériovým portem a další.* OpenSDA tedy poskytuje sériovou a debug komunikaci mezi USB hostem (připojeným přes microUSB) a cílovým mikrokontrolérem. Hardwarový obvod je založen na MCU řady Kinetis K20 s vnitřní pamětí programu o velikosti 128 KB a integrovaným řadičem USB. Na obr. 16 je zobrazeno blokové schéma tohoto rozhraní. [3]



Obr. 16: *Blokové schéma OpenSDA rozhraní* [3]

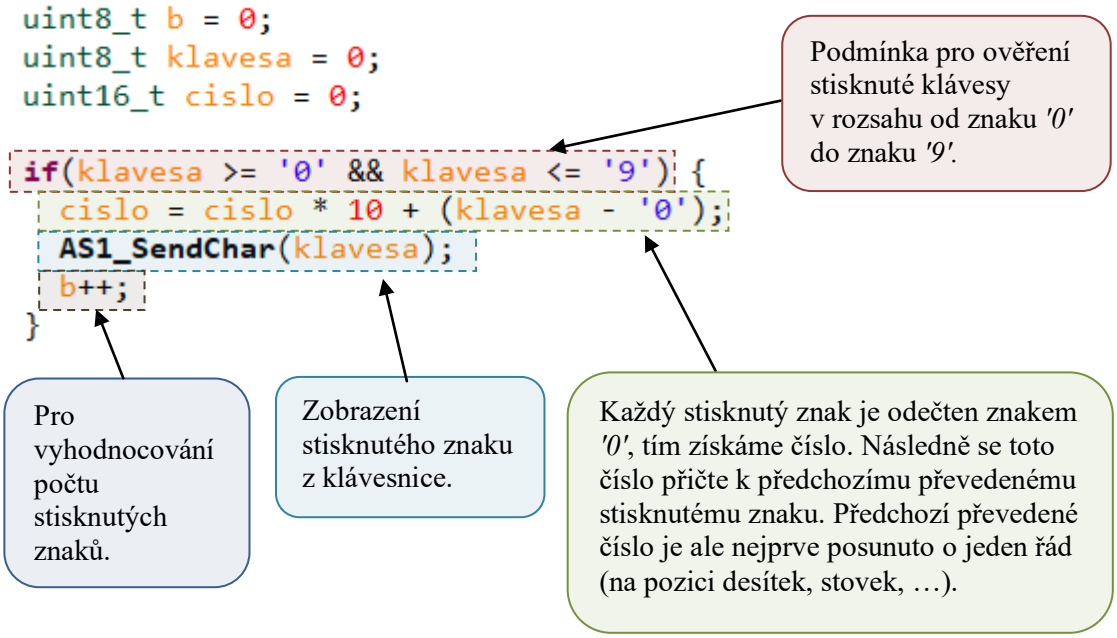
Pomocí jmenovaného rozhraní vytváříme tzv. virtuální připojení. Zjednodušeně to znamená, že připojením MCU k PC prostřednictvím OpenSDA a USB kabelu se vytvoří spojení, které PC vnímá jako COM port, naopak MCU jej pokládá za běžné UART připojení.

2.2.3 Způsob vypracování a ověření výsledků

Vypracování

Student postupuje podle návodu, kde jsou popsány funkce komponent AsynchroSerial, XFormat a Utility, které student ke své práci využívá. Dále je proveden nastavením parametrů přenosu prostřednictvím komponenty AsynchroSerial, jenž umožňuje jednoduché nastavení vstupního a výstupního bufferu vybraného UART modulu mikrokontroléru, zvolit parametry komunikace (velikost přenášeného datového slova, nastavení počtu Stop bitů a také nastavení paritního bitu), komponenta umožňuje jednoduše a hlavně správně nakonfigurovat příslušné piny, kterými se bude uskutečňovat komunikace bez nutnosti vlastního přístupu k jednotlivým registrům.

Studentovi je názorně ukázán způsob praktické realizace převodu stisknutého znaku z klávesnice na několikamístné číslo, viz níže.



Mikrokontrolér MK64FN1M0VLQ12 nám poskytuje celkem 6 UART modulů. Na naší desce máme přístup k pěti z nich, kde UART0 je připojeno na rozhraní RS422 a UART3 na rozhraní RS485. Hodinový signál pro UART0 a UART1 moduly je distribuován z *core/system clock*, tyto zdroje umožňují větší výkon modulů. Hodinový signál ostatním modulům je distribuován z *bus clock*. Zvolený hodinový signál všech modulů je upravován pomocí děliček kmitočtu. Výsledkem je tedy maximální přenosová rychlost 1/16 zvoleného zdroje hodinového signálu. Pro komunikaci přes OpenSDA nám slouží modul UART4. [6]

Používané funkce komponent

AsynchroSerial (AS1:AsynchroSerial)

- *AS1_GetCharsInTxBuf()* = pro zjištění zaplnění výstupního bufferu
- *AS1_RecvChar()* = příjem dat o velikosti 1 byte
- *AS1_SendChar()* = vysílání dat o velikosti 1 byte
- *AS1_RecvBlock()* = příjem řetězce dat
- *AS1_SendBlock()* = vysílání řetězce dat

XFormat (XF1:XFormat)

- *XF1_xsnprintf()* = formátování řetězce do pole prvků

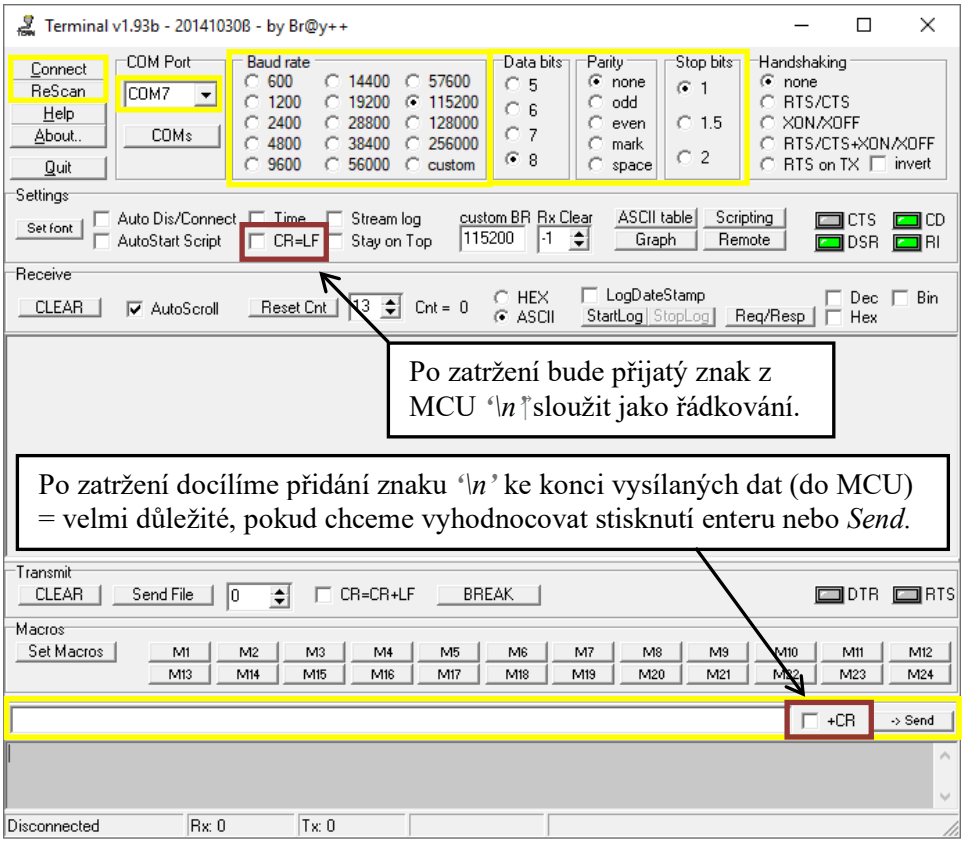
Utility (UTIL1:Utility)

- *UTIL_Num16sToStr()* = převedení 16bitového čísla na řetězec znaků s uložením do pole

Ověření výsledků

Pro ověření správnosti napsané aplikace se student naučí, jak pracovat s programem Terminal v1.93b. Jedná se o komunikační terminál, který je na internetových stránkách volně dostupný. Umožňuje nám široké možnosti nastavení přenosové rychlosti ve velkém rozsahu a nastavení komunikace pro příjem, vysílání dat a obsluhu sériových rozhraní. Poskytuje nám možnosti posílat data v různých formátech (ASCII, HEX a jiné).

Na obr. 17 je zobrazení programu s popisem a vysvětlením základních bloků nastavení, jejichž funkci je potřeba pochopit pro ověření správnosti napsané aplikace.



Obr. 17: *Terminal v1.93b pro čtení, ukládání a zasílání dat pomocí sériových portů*

Connect … navázání komunikace s připojeným zařízením (mikrokontrolérem)
ReScan … opakování skenování pro nalezení nově dostupných sériových portů
COM Port … výběr portu sériového rozhraní
Baud rate … volba přenosové rychlosti (musí se shodovat u obou zařízení, které spolu komunikují)
Data bits, *Parity*, *Stop bits* … nastavení parametrů přenosu dat

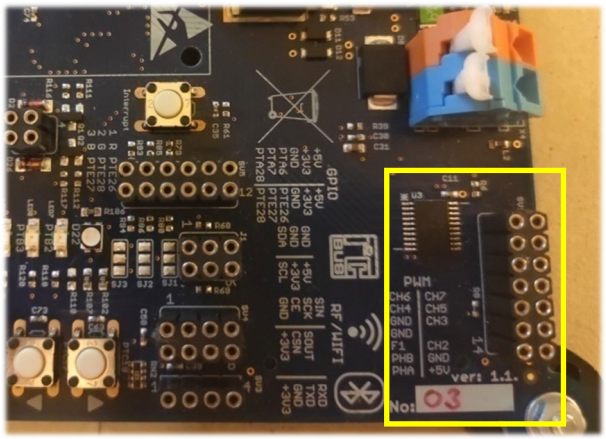
Vysílání se uskutečňuje zapsáním znaku nebo řetězce do spodního vyznačeného bloku s následným stiskem tlačítka *Send*.

2.3 Generování PWM signálu

2.3.1 Představení laboratorní úlohy

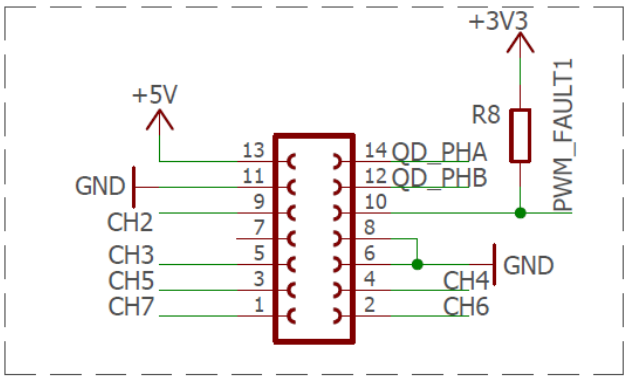
Student se seznámí s principy generování PWM signálu. Jeho prvním úkolem je tento signál generovat softwarově pomocí periodického přerušení časovačem. Střída signálu bude periodicky narůstat a klesat. Změnu střídy bude pozorovat na měnícím se jasu led.

Dalším úkolem je generovat PWM signál hardwarově využitím speciálního časovacího modulu FTM3 a komponenty PWM. Prostřednictvím sériového kanálu a programu Terminal, student vytvoří ovládací menu, které bude uživateli sloužit pro ovládání šířky signálu třemi odlišnými způsoby. Kanály FTM3 modulu jsou na desce s mikrokontrolérem vyvedeny přes linkový budič na 14pinový konektor.



Obr. 18: *Obvod pro generování hardwarového PWM s linkovým budičem na desce*

Obr. 19 zobrazuje schéma zapojení PWM konektoru, z něhož jsou vyvedeny jednotlivé kanály modulu časovače FTM3. Na konektoru jsou vyvedeny také piny QD_PHA a QD_PHB, které slouží speciálně jako vstupní piny pro zpracování výstupních signálů z rotačního enkodéru.



CH2	FTM3_CH2
CH3	FTM3_CH3
CH4	FTM3_CH4
CH5	FTM3_CH5
CH6	FTM3_CH6
CH7	FTM3_CH7

Obr. 19: *Popis rozmístění kanálů modulu FTM3 na konektoru*

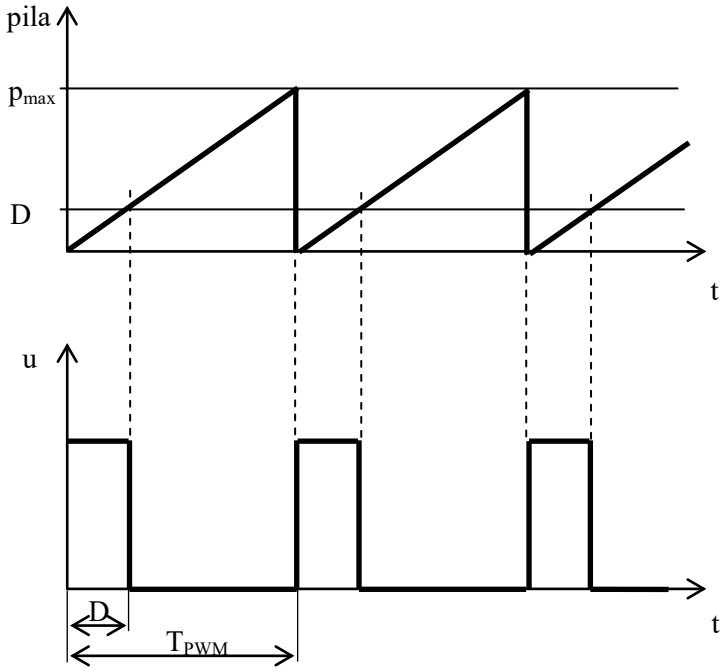
2.3.2 Pulzní šířková modulace

Pulzní šířková modulace je diskrétní modulace využívaná pro přenos analogového signálu prostřednictvím dvoustavového signálu. V elektronice dnes nachází široké uplatnění. Využívá se zpravidla pro řízení rychlosti motorů, kdy nastavením šířky PWM signálu se reguluje střední hodnota napětí přiváděného na motor, řízení světelných zařízení, klimatizace apod. Používá se mimo jiné jako levná náhrada za D/A převodník, kdy šířka signálu je určena vstupní digitální hodnotou přiváděnou na převodník a výsledkem je opět změna střední hodnoty napětí. [10]

Princip spočívá v generování signálu s konstantní frekvencí a proměnnou šířkou impulzu (*duty cycle*), jinak řečeno, řídí se doba, kdy se výstupní signál nachází ve stavech log. 1 a log. 0. Tato modulace je v MCU velice často založena na porovnávání výstupní hodnoty registru časovače pracujícího v režimu *output compare.*

Generování softwarového PWM

Vytvoření jednoduchého softwarového PWM zachycuje obr. 20, na kterém bude popsán způsob jeho vytvoření. Princip spočívá ve vytvoření pilovitého signálu inkrementací hodnoty proměnné pila za přesně danou dobu, kdy po překročení maximální hodnoty označené p_{max} dojde k vynulování jejího obsahu a k překlopení výstupního signálu na log 1. Čas inkrementace proměnné pila můžeme nastavit přerušením časovače. Perioda našeho PWM signálu bude rovna době inkrementování proměnné, tedy době kdy se hodnota proměnné pila pohybuje od 0 do překročení maximální hodnoty p_{max}. Šířku výstupního PWM signálu nastavujeme hodnotou v proměnné D (*duty cycle*). Bude-li platit podmínka pila > D, bude se výstupní signál nacházet v log. 0.



Obr. 20: *Princip generování softwarového PWM*

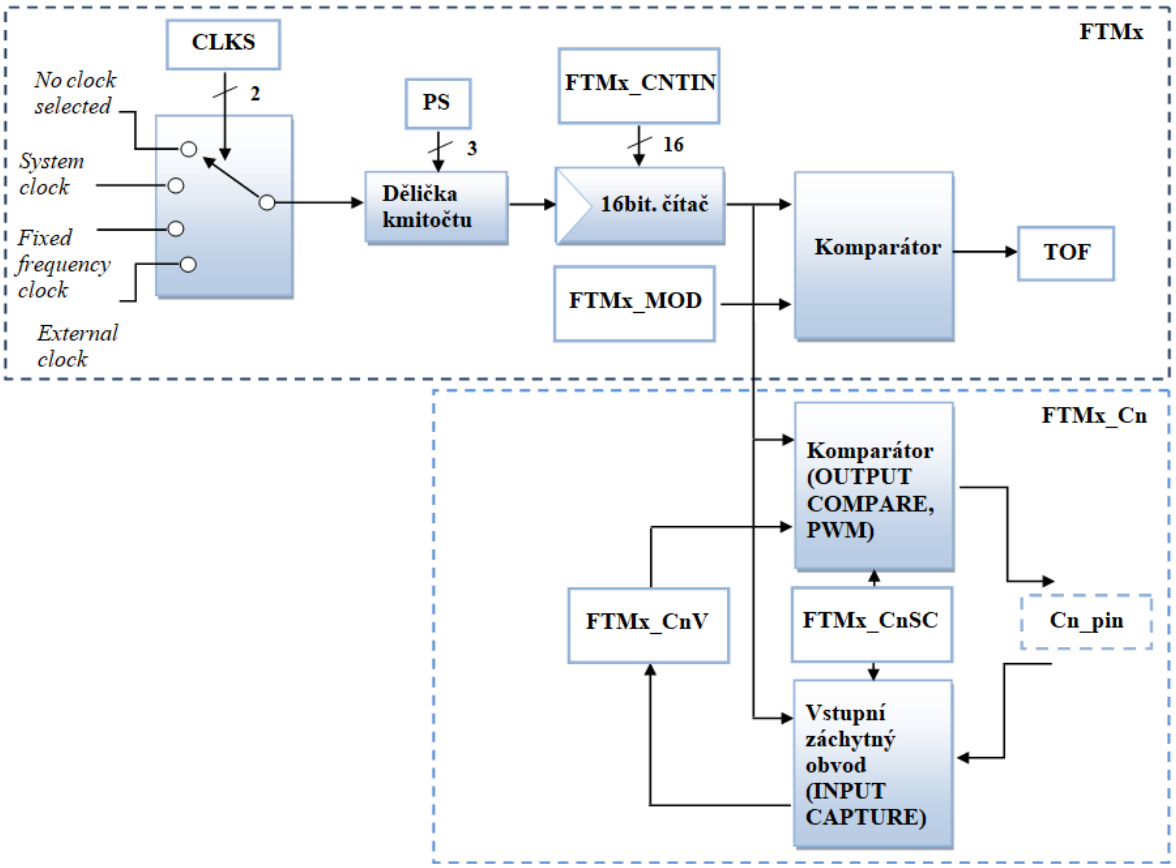
Hardwarové PWM pomocí časovače FlexTimer (FTM) [11]

Jedná se o časovací moduly speciálně navržené pro řízení měničů pomocí PWM. Mikrokontrolér je vybaven celkem dvěma osmi kanálovými moduly FlexTimer a dvěma dvou kanálovými moduly FlexTimer.

Je možné jej použít v režimu *input capture*, *output compare* a také právě ke generování hardwarových PWM signálů. Každý jednotlivý FTM modul využívá ke své funkci 16bit čítač s programovatelnou počáteční a konečnou hodnotou. FTM poskytuje široké možnosti generování PWM, jsou jimi např. výběr směru čítání čítače (nahoru nebo nahoru a dolů), podle našich požadavků na řízení, vložení *deadtime*, inicializace a řízení polarity. Nastavení zmíněných režimů se provádí v příslušných registrech vybraného modulu, v našem případě však není potřeba vědět, jaké registry se nastavují, konfiguraci registrů totiž za nás udělá nástroj Processor Expert.

Funkce a možnosti použití FTM: [2]

- jako časovač nebo PWM generátor
- použití v režimu *input capture* nebo *output compare*
- každý modul je schopen generovat maximálně 8 PWM signálů
- zarovnání PWM signálu doprava, doleva nebo zarovnání na střed s funkcí *deadtime*
- umožňuje fázový posun PWM signálu



Obr. 21: *Zjednodušené blokové schéma modulu FTM (podle* [1, 2] *)*

Základní princip funkce generování PWM: [2]

Pro každý modul FTM máme možnost vybrat různé zdroje hodinového signálu. Jednotlivé moduly jsou vybaveny vnitřní děličkou pro úpravu kmitočtu vybraného hodinového signálu. Každý FTM je vybaven 16bitovým čítačem, který se přednastavuje na hodnotu určenou v registru FTM_CNTIN. Následně čítač čítá vnitřní impulsy, jejichž perioda byla nastavena právě děličkou kmitočtu. Ten čítá až po dosažení hodnoty v registru FTM_MOD. Tak je přesně nastavená požadovaná perioda výstupního PWM signálu. Pro nastavení šířky signálu je FTM vybaven registrem FTM_CnV. Po dosažení hodnoty nastavené v registru FTM_CnV se změní stav na výstupním kanálu FTM modulu, který jsme předem museli nakonfigurovat na danou funkci PWM. Výstupní kanál tedy změní logický stav.

Popis funkce jednotlivých registrů [2]

- FTMx_SC

Registr obsahuje 2 bity CLKS pro výběr jednoho ze čtyř hodinových signálu. Jsou jimi *system clock*, *fixed frequency clock*, *external frequency clock* a možnost výběru žádného zdroje, tedy *no clock selected*, tehdy není čítač funkční. Součástí registru je bitové pole PS pro nastavení dělícího poměru kmitočtu vybraného zdroje hodinového signálu. Dělící poměr je možné nastavit 1/2/4/8/16/32/64/128. Dále bitové pole CPWMS pro výběr režimu čítání 16bit. čítače (čítání nahoru nebo čítání nahoru a následně dolů). Nakonec zde najdeme TOF bit indikující přetečení čítače.

- FTMx_CNTIN

Nastavení počáteční hodnoty 16bit. čítače.

- FTMx_MOD

Nastavení koncové hodnoty 16bit. čítače. Po dosažení dojde k přenastavení obsahu čítače hodnotou v registru FTMx_CNTIN.

- FTMx_CnSC

Registrem nastavujeme jeden z módů jednotlivého kanálu FTM modulu.

Režimy kanálů [2]

- *Input Capture* = zachycení změny stavu na pinu Cn_pin pracujícího ve vstupním režimu. V tomto módu 16bitový čítač čítá tak dlouho až se na vstupním pinu časovače detekuje hrana (vzestupná, sestupná nebo vzestupná i sestupná hrana), následně se obsah čítače nahraje do FTMx_CnV registru, s hodnotou můžeme následně pracovat, např. můžeme zjistit dobu mezi dvěma stavy. To můžeme vyžít pro zjištění doby trvání určitého logického stavu na vstupu a zjistit tak mj. šířku PWM signálu.
- *Output Compare* = porovnává se obsah 16bitového čítače s obsahem registru FTMx_CnV, při rovnosti se vykoná změna na výstupním pinu (nastavení výstupu na logickou 1 nebo 0, negace logického stavu).
- Generování PWM signálů

2.3.3 Způsob vypracování a ověření výsledků

Vypracování

Rovněž zde student postupuje dle návodu, ze kterého se dozví o možnostech použití komponenty PWM, kterou student ke své práci využívá. Komponenta umožňuje jednoduché nastavení parametrů hardwarového PWM signálu generovaného zvoleným kanálem FTM modulu, umožňuje také nastavení periody signálu, nastavení počáteční úrovně a nastavení inicializace. Jelikož se výběr a řízení (použitím ovládacího menu) generovaného signálu provádí prostřednictvím rozhraní OpenSDA, je zde taktéž nutné pracovat s komponentou AsynchroSerial a XFormat. Jako komunikační terminál slouží Terminal v1.93b, pomocí kterého se výběrové menu bude uživateli zobrazovat.

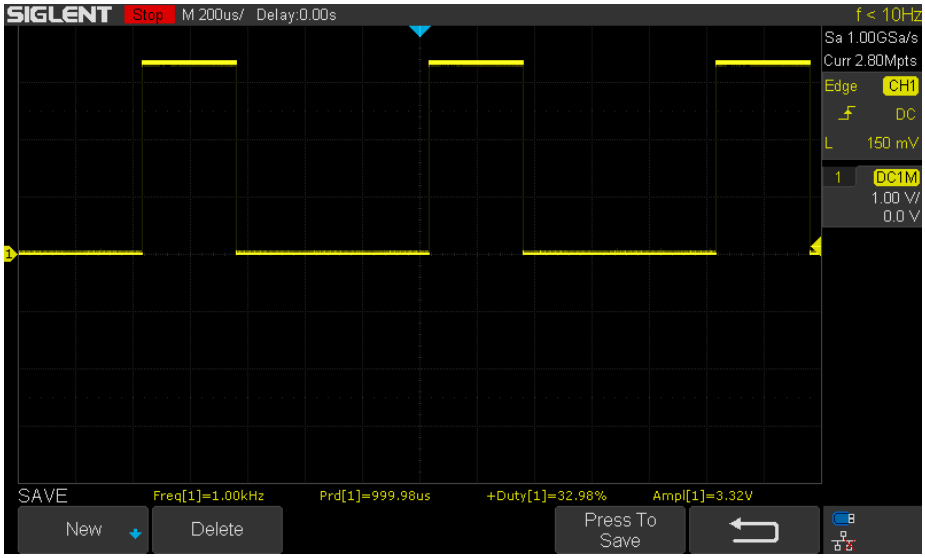
Používané funkce komponent

PWM (PWM1:PWM)

- *PWM1_SetRatio16()*= nastavení procentní šířky PWM signálu 16bitovou hodnotou
- *PWM1_SetDutyMS()*= vyjádření nastavené procentní šířky v čase

Ověření výsledků

Po vytvoření programu je možné generovaný signál analyzovat a vyjádřit se tak k možnostem softwarového a hardwarového PWM.



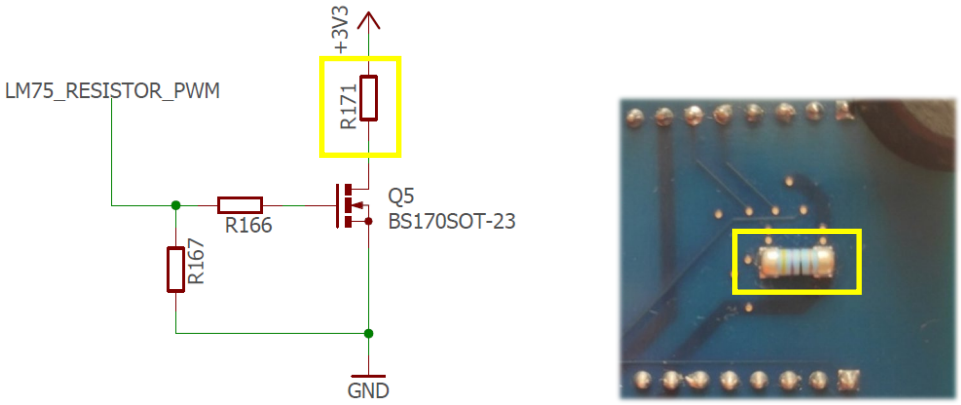
Obr. 22: *Zobrazení generovaného PWM signálu, nastavená šířka 33 %*

2.4 Komunikace mikropočítače s teplotním čidlem přes sběrnici I2C

2.4.1 Představení laboratorní úlohy

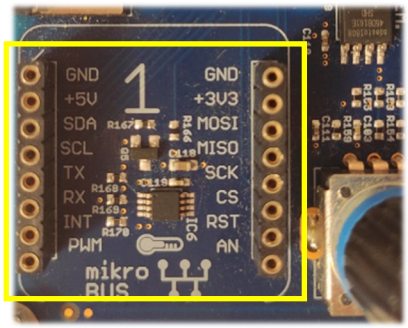
Student se seznámí s principy sériové komunikace po sběrnici I2C a sám vytvoří aplikaci pro řízení komunikace s obvodem teploměru LM75AD, kterým se bude měřit teplota na odporu R171. Odpor je připojován a odpojován od napájení pomocí tranzistoru MOSFET s indukovaným kanálem. Spínáním tranzistoru se ve výsledku řídí teplota na odporu na zvolenou úroveň, jako zpětná vazba slouží právě teplotní senzor, který nám podává informace o skutečné teplotě na odporu.

Obvod LM75AD je teplotní senzor převádějící teplotu na digitální hodnotu, s kterou je pak možné pracovat. Pro převod z analogové hodnoty do digitální využívá 11bitový Sigma-delta A/D převodník. Senzor obsahuje celkem 4 registry, jsou jimi konfigurační registr (*Conf register*), kterým se nastavuje operační mód senzoru, dále registr pro uchování převedené teploty v digitální podobě (*Temp register)* a registry (*Tos and Thyst*) pro nastavení maximální teploty a hystereze v příslušném módu senzoru. [13]

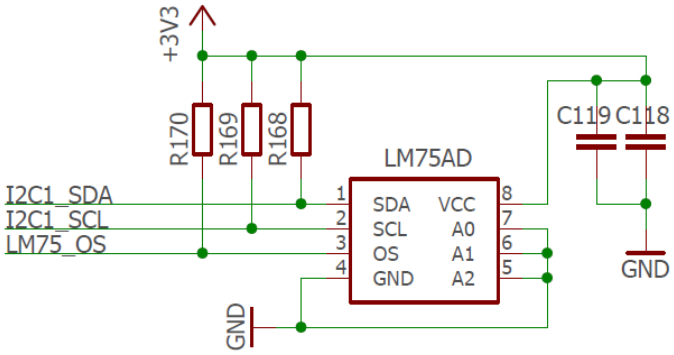


Obr. 23: *Obvodové schéma pro spínání odporu R171 (vlevo) se zobrazením odporu na zadní desce (vpravo)*

Pro správné nastavení komunikace a její řízení slouží komponenta GenericI2C, konfigurace pinu pro spínání tranzistoru provádíme komponentou BitIO. Student je provázen nastavením parametrů přenosu podle datasheetu od výrobce teplotního senzoru.



Obr. 24: *Zapojení teplotního senzoru LM75AD s konektory na desce*

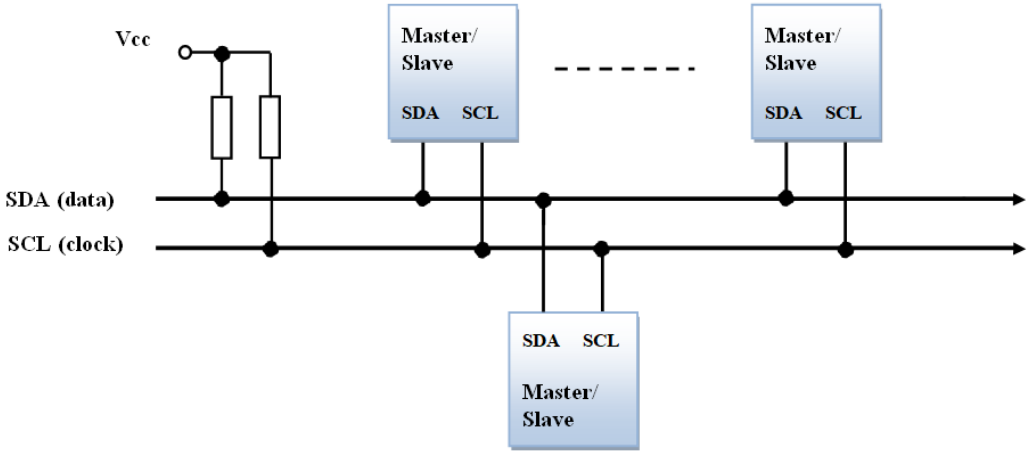


I2C1_SDA	PTC11
I2C1_SCL	PTC10
LM75_OS	PTC8

Obr. 25: *Obvodové schéma zapojení teplotního senzoru LM75AD (vlevo) s popisem vyvedených pinů z mikrokontroléru (vpravo)*

2.4.2 Sběrnice I2C [12]

Sběrnice byla vyvinuta společností Philips v 80. letech z důvodu snížení výrobních nákladu. Je založena na jednoduchosti, aniž by to znamenalo pokles na rychlost komunikace. Jde v dnešní době již o rozšířený standard ve většině oblastí současné elektroniky, proto je její pochopení zásadní.



Obr. 26: *Blokové schéma zapojení zařízení na I2C sběrnici*

Vlastnosti sběrnice

Je polo duplexní sběrnice s maximální rychlostí přenosu 3,4 Mbit/s. Logické úrovně jsou odvozeny z napájecího napětí sběrnice, log. 1 je určena od 70% napájecího napětí a log. 0 do 30% napájecího napětí. Všechna zařízení se připojují pouze dvouvodičově. Přenos dat probíhá po SDA (*serial data line*), synchronizace přenášených dat je zprostředkována pomocí SCL (*serial clock line*). Každé připojené zařízení má svou jedinečnou adresu, kterou je softwarově identifikováno zařízením Master, kdy množství připojených zařízení je limitováno počtem bitů adresování a celkovou kapacitou (400 pF) na sběrnici. Sběrnice umožňuje obousměrný přenos dat, kdy v daném okamžiku probíhá

přenos vždy mezi dvěma zařízeními na sběrnici. Zařízení mohou být typu Master nebo Slave. Na sběrnici může být připojeno více zařízení jak typu Slave, tak typu Master. Pouze Master má však možnost zahajovat a ukončovat komunikaci. Slave je podřízený Masteru a je tedy povinen vyčkávat na zahájení komunikace, účastní se komunikace jen po zavolání jeho adresy. Zařízení na sběrnici pracují v jednom ze dvou režimů, vysílání nebo přijímání dat. Master generuje hodinový synchronizační signál a může data vysílat a také přijímat. Slave je tedy Masterem adresováno a může pracovat buď jako vysílač nebo jako přijímač, avšak neplatí to vždy, používáme-li např. řadič displeje, ten může zastávat pouze funkci příjemce dat pro zobrazení znaků na displeji. Během adresování Masterem vysílanou adresu přijímají všechna připojená Slave zařízení.

Hardwarové nastavení

Každá sběrnice I2C musí být vybavena *pull-up* odpory hodnoty od 2 kΩ do 10 kΩ na obou vodičích (SDA a SCL), ty jsou připojeny na kladné napájení (+3 V nebo +5 V). Tím se docílí trvalého nastavení klidové úrovně log. 1 na sběrnici. Piny každého připojeného zařízení jsou nastaveny s otevřeným kolektorem, pomocí spínání tranzistoru pak zařízení nastavují logické úrovně a komunikují tak po společné I2C sběrnici. Vysílaná data jsou získávána zesilovači s vysokou vstupní impedancí, kterými jsou zařízení s I2C modulem na svých pinech vybavena. Uživatel má někdy možnost nastavit adresu připojeného zařízení. To nám přináší výhodu, obzvlášť máme-li používaná zařízení na společné sběrnici se stejnou adresou. Využíváme-li např. 7bitové adresování, pak první čtyři bity jsou pevné a tři nejméně významné je možné libovolně nastavit.

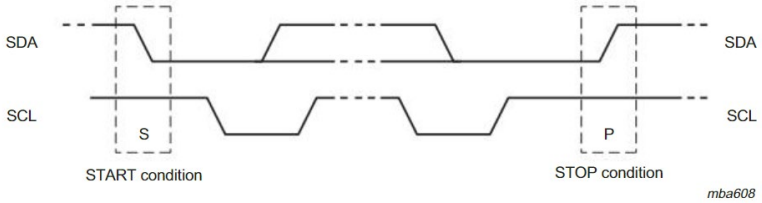
Použití I2C má své nesporné výhody. Dále budou zmíněny jen některé:

- Protokol komunikace je poměrně jednoduchý.
- Využívá softwarové adresování.
- Vyžaduje pouze dvouvodičové spojení pro komunikaci mezi zařízeními = snížení ceny a zastavěného prostoru na desce spojů, menší počet použitých pinů integrovaných obvodů, jednoduchá diagnostika a řešení chyb.
- Flexibilita = jednoduchá změna počtu připojených zařízení, odebrané nebo připojené zařízení na již existující I2C sběrnici neovlivňuje celkovou funkci sběrnice.
- Velmi známý protokol, rozšířen u výrobců integrovaných obvodů a používán v mnoha ostatních zařízení = zrychlení vývoje softwaru, konfigurace a řízení přenosu díky dostupnosti mnoha již napsaných knihoven.
- Umožňuje režim multi-Master, tedy připojení většího počtu zařízení typu Master s funkcí arbitráže.

Princip komunikace

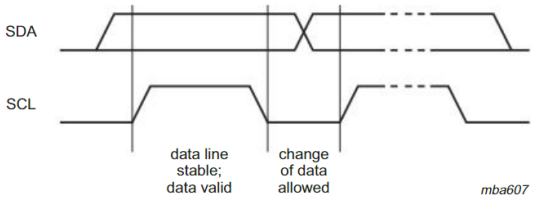
V klidovém stavu se sběrnice nachází v log. 1. Komunikace je zahájena Masterem vysláním Start bitu. Ten se vyznačuje změnou klidového stavu na log. 0 na SDA vodiči. Tato změna se provádí během SCL nacházejícím se v log. 1. Ihned po vyslání Start bitu dotyčný Master přebírá kontrolu nad sběrnicí, jiný Master tedy již nemůže zahajovat nový přenos. Nové zahájení komunikace, jakéhokoliv

zařízení, je umožněno až po uvolnění sběrnice vysláním Stop bitu, tedy změnou log. 0 na log 1 na SDA, během níž se SCL nachází ve stavu log. 1.



Obr. 27: *Vyslání Start a stop bitu zařízením typu Master* [4]

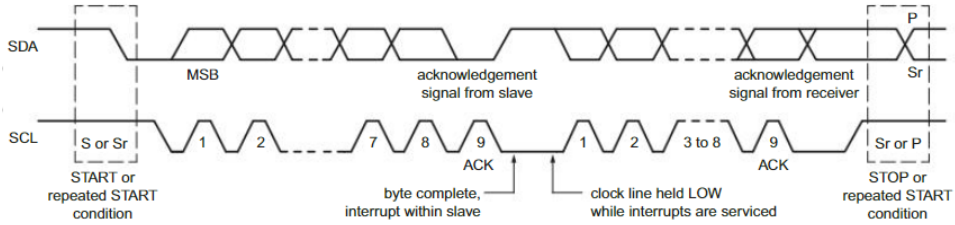
Master po získání kontroly vysílá po SCL synchronizační hodinový signál, od něhož je odvozena rychlost přenosu. Změna logického stavu po SDA probíhá jedině tehdy, nachází-li se SCL ve stavu log. 0. Každý bit dat je přenášen po jednotlivém hodinovém impulsu.



Obr. 28: *Přenos dat* [4]

Může také nastat situace, kdy se o kontrolu nad sběrnicí uchází více než jeden Master. Tehdy se uplatňuje arbitráž. Po vyslání Start bitu každý Master začíná adresováním Slave zařízení na sběrnici, tehdy se porovnává stav vysílaných bitů adresy. Princip je následující, Master 1, u kterého je poprvé zaznamenán stav log. 0 a v témže okamžiku vysílá Master 2 stále log. 1, vyhrává arbitráž Master 1 a získává tak kontrolu nad sběrnicí. Master 2 pak musí vyčkat na uvolnění sběrnice Masterem 1. Během arbitráže se hodinový signál na SCL skládá z kombinace hodinových signálů generovaných oběma Mastery.

Po Start bitu od Masteru následuje vyslání 7bitové či 10bitové adresy zařízení, které se má komunikace zúčastnit. Tato adresa je přijata všem připojeným zařízením. Po adrese následuje bit pro čtení (log. 1) či zápis (log. 0). Po 9. bitu převezme iniciativu adresované Slave zařízení, tehdy Master SDA uvolní a během 10. hodinového impulsu Slave s požadovanou adresou stáhne logickou úroveň SDA na log. 0. Data jsou přenášena osmibitově a všechna vysílaná data jsou přijímacím zařízením potvrzována přenášeným devátým bitem. Aby se potvrzení opět mohlo uskutečnit, vysílací zařízení uvolní SDA během 9. hodinového impulsu, přijímač následně stáhne logickou úroveň na log. 0 pro potvrzení nebo ponechá log. 1 pro indikaci negativního potvrzení. Pokud Slave nestačí vysílat či přijímat data na sběrnici s frekvencí přenosu nastavenou Masterem, pak Slave podrží SCL na log. 0. Tím dá najevo Masteru, že má s přenosem dalších bitů dat počkat do uvolnění SCL. Je-li vyžadován požadavek přenosu více než pouze jednoho přenášeného bajtu dat, vysílá se opakovaný start.



Obr. 29: *Zobrazení průběhu celé komunikace* [4]

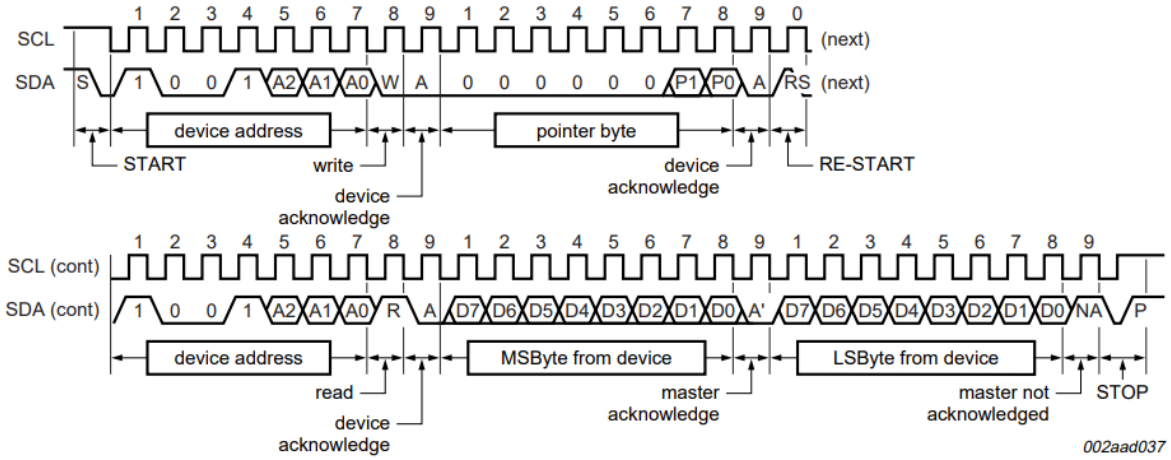
2.4.3 Způsob vypracování a ověření výsledků

MCU je vybaven třemi I2C moduly (I2C0, I2C1 a I2C2), které jsou připojeny k *bus clock* s podporovanou maximální přenosovou rychlostí až 1 Mbit/s. Maximální délka sběrnice a počet připojených zařízení je omezeno maximální kapacitou sběrnice 400pF. Moduly jsou vybaveny kontrolními, stavovými a datovými registry pro nastavení a řízení přenosu dat. Student nemusí tyto zmíněné registry složitě nastavovat, jelikož celé nastavení komunikačního protokolu je prováděno pomocí nástroje PE v grafickém okně.

Pro čtení dat ze senzoru LM75AD je potřeba dodržovat následující pravidla: [13]

1. SDA a SCL se musí nacházet v log. 1, tedy komunikační linka není používaná jinými obvody na I2C sběrnicí.
2. Master, tedy zařízení, které chce navázat se senzorem (Slave) kontakt, poskytuje hodinový signál na SCL. Celkem vygeneruje 9 hodinových pulzů pro každý přenášený byte dat. Devátý hodinový impuls slouží pro potvrzení přenosu (A) od senzoru. To se uskuteční uvolněním sběrníce Masterem (log. 1). Slave následně změní stav na log. 0 pro potvrzení nebo ponechá stav nezměněný pro indikaci chyby přenosu dat.
3. Vždy během log. 1 hodinového signálu na SCL se signál na SDA nachází ve stabilní úrovní (v log. 0 nebo log. 1). Změna stavů se provádí pouze během log. 0 na SCL. Vyjímka platí při vysílaní START a STOP.
4. Generování START probíhá změnou klidové úrovně (log. 1) na log. 0 během SCL nacházejícím se ve stavu log. 1.
5. Komunikace vyžaduje RE-START (RS), jelikož chceme nejen data zapisovat (*pointer byte*), ale také číst (naměřená teplota).
6. Pro ukončení komunikace Master generuje STOP, změnou stavu na SDA z log. 0 na log. 1, během SCL nacházejícím se v log. 1.
7. Pred vysláním STOP je potřeba zajistit vyslání NA (*not acknowledge*) bitu, což je okamžik uvolnění sběrnice oběma zařízeními účastnícím se komunikace.
8. Je vyžadováno vyslání jak čtecího bitu (W), tak také zapisovacího bitu (R).
9. Po příjmu prvního bajtu dat ze senzoru, musí Master odpovědět porvrzením (A´), tedy změnou stavu na SDA log. 0.
10. Master má plnou kontrolu nad sběrnici během zápisu do registrů senzoru, sběrnici musí však také v danou dobu uvolnit pro vyslání dvou bajtů dat senzorem.

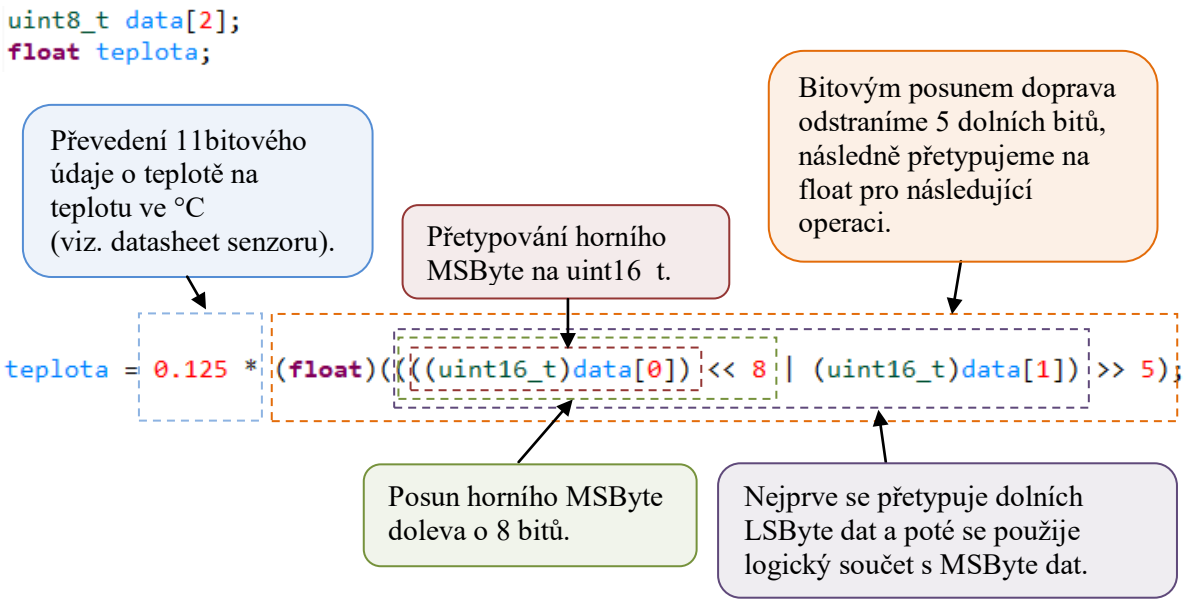
Obr. 30 zachycuje průběh komunikace mezi Masterem *(MCU)* a zařízením Slave *(senzorem)* s vyznačením vyžadovaných stavů.



Obr. 30: *Zobrazení komunikačního protokolu pro čtení dvou bajtů dat z teplotního senzoru* [13]

Součástí návodu je popis použití komponenty GenericI2C, kterou student ke své práci využívá. Komponenta umožňuje jednoduché nastavení parametrů přenosu dat a komunikačního protokolu pro zvolené zařízení. Mezi používaná nastavení můžeme zařadit nastavení adresování, zvolení pinů pro alternativní funkci SDA a SCL, volba frekvence a časových parametrů hodinového signálu. Komponenta rovněž využívá události přerušení během datového přenosu. Dále je využívána, pro spínání tranzistoru a tedy změnu teploty na měřeném odporu, komponenta BitIO. Ta nám poskytuje volbu pinu pro výstupní funkci.

Studentovi je ukázána možnost realizace převodu získané teploty ze senzoru na údaj o skutečné teplotě v °C, s kterou je možné již pracovat, viz níže.



Používané funkce komponent

GenericI2C (GI2C1:GenericI2C)

- *GI2C1_ReadAddress()* = čtení dat ze zařízení připojeného na společné I2C sběrnici

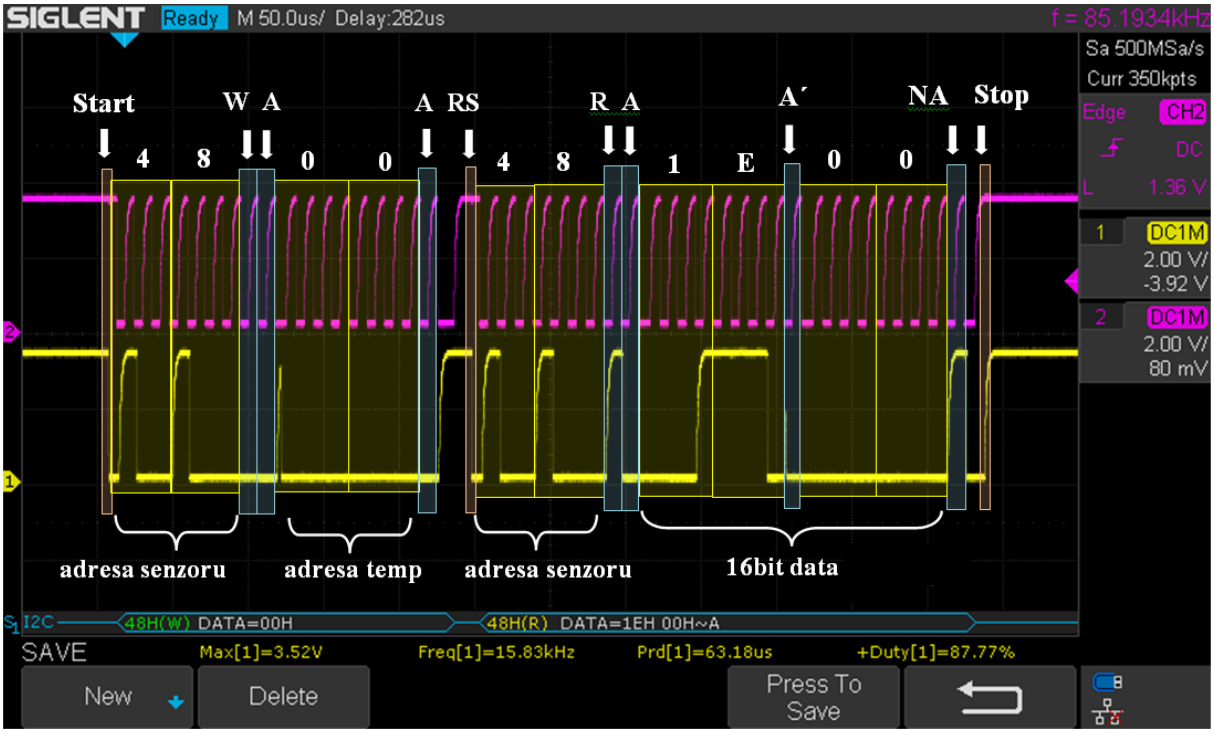
BitIO (Bit1:BitIO)

- *Bit1_SetVal()* = nastavení zvoleného výstupního pinu na logickou úroveň 1
- *Bit1_ClrVal(void)* = nastavení zvoleného výstupního pinu na logickou úroveň 0

Ověření výsledků

Po úspěšném napsání programu student analyzuje, pomocí dekódovací funkce osciloskopu, komunikaci po I2C lince. Následně je schopen rozlišit jednotlivé bity sériové komunikace na SDA. Student si tak skutečně ověří informace, které se dozvěděl z teoretického rozboru, tedy informace o podmínkách, které vedou k úspěšné komunikaci s připojeným zařízením na sběrnici. Tak se přesvědčí o funkci jednotlivých stavů, které předcházejí zahájení a ukončení přenosu dat, potvrzení přenosu, rozlišení zápisu či čtení přenášených dat. Prakticky se přesvědčí o funkci opakovaného startu, který je během správné komunikace vyžadován. Na obr. 31 je zobrazena dekódovaná komunikace s vyznačením jmenovaných stavů spolu s rozlišením přenosu adresy připojeného zařízení (0x48h), adresy do paměti zařízení (0x00h) a nakonec také přijímaná 16bitová data naměřené teploty senzorem.

Student je tak, po provedení laboratorní úlohy, schopen aplikovat nabyté poznatky pro další aplikace, získá základ pro uskutečnění komunikace s kterýmkoliv zařízením prostřednictvím I2C.



Obr. 31: *Zobrazení dekódované komunikace po sběrnici I2C mezi MCU a teplotním senzorem*

2.5 Zobrazování na displeji pomocí SPI rozhraní

2.5.1 Představení laboratorní úlohy

Cílem je seznámit studenta s principem komunikace po SPI sběrnici s použitím periferních modulů MCU a aplikovat tyto poznatky do řízení displeje typu Nokia 3310 s řadičem PDC8544. Student má za úkol vytvořit program pro vypsání textu a také tabulky s proměnným počtem řádků a sloupců na displeji. Pokud studenta problematika zobrazování více zajímá, má možnost dále vypracovat bonusové zadání, jehož úkolem je zobrazení libovolného obrázku nebo animace. Studentovi je poskytnut návod pro vypracování úlohy s vysvětlením konkrétních kroků postupu.

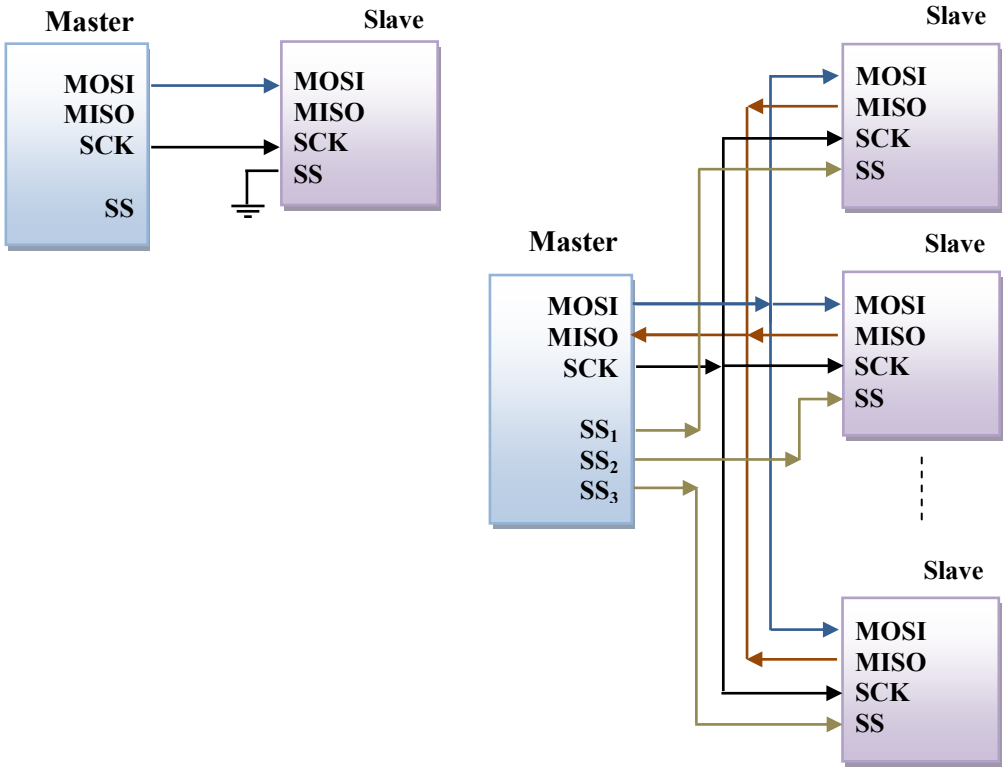
Student ke své prácí využívá komponentu PDC8544, která je speciálně určena pro práci s řadičem displeje jmenovaného typu, a také GDisplay sloužící jako grafický ovladač pro zobrazování jednoduchých obrazců a obrázků. Součástí návodu jsou také popisy použití používaných funkcí komponent pro splnění zadání.

Obr. 32: *Vykreslení tabulky, textu a obrázku na displeji*

2.5.2 SPI sběrnice

Základní vlastnosti sběrnice [1, 14]

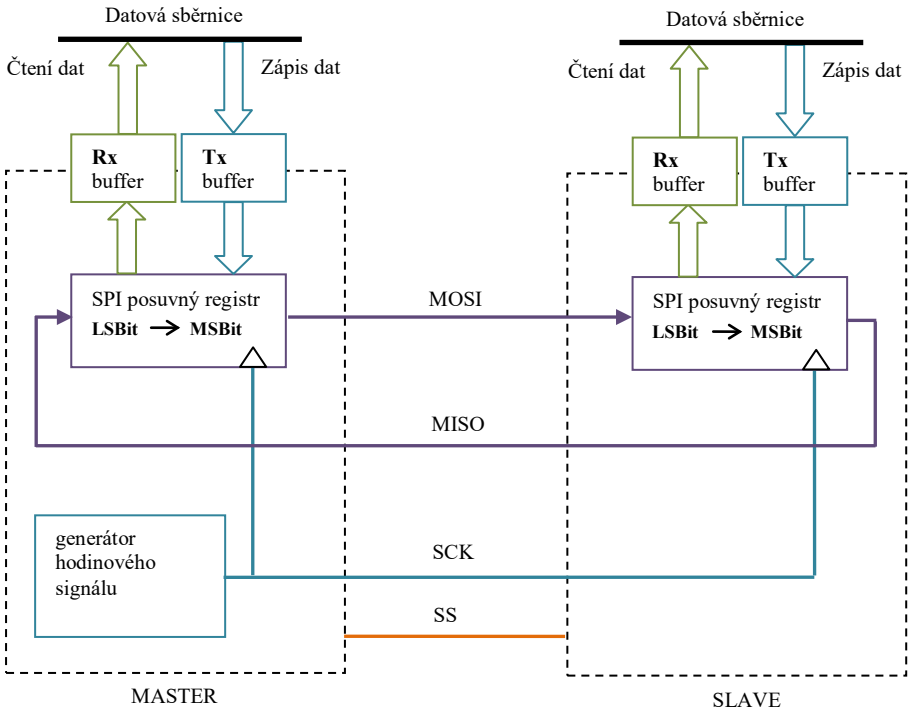
Řadí se mezi synchronní USART. SPI komunikace probíhá mezi zařízením Master a jednoho nebo i většího množství připojených zařízení pracujících v řežimu Slave. Pokud vytváříme SPI spojení s více než jedním zařízením, je nutné komunikaci vytvořit připojením čtyř vodičů. Pokud se jedná pouze o jedno připojené Slave zařízení na společné sběrnici, postačí pouze vodiče dva. Příkladem takového spojení je MCU s displejem (obr. 33 vlevo). Data jsou vysílána po vodiči MOSI a řadičem displeje přijímána pro konečné zobrazení. Jelikož SPI umožňuje komunikaci mezi řídicím zařízením a více podřízenými obvody, nachází dnes široké uplatnění pro komunikaci s pamětí, displejem, s A/D převodníky a s dalšími přídavnými periférierni k mikrokontroléru. Data je možné přenášet zároveň oběma směry, SPI je tedy plně duplexní. Používá se tam, kde požadujeme dosažení vysoké přenosové rychlosti (frekvence až 70 MHz) na krátké vzdálenosti.



Obr. 33: *Připojení zařízení ke společné SPI sběrnici s minimálním připojením (vlevo), většího počtu Slave zařízení (vpravo)*

Přenos dat po SPI sběrnici probíhá po dvou vodičích, nazývají se MOSI (*Master out Slave in*) a MISO (*Master in Slave out*). Každé zařízení Slave je vybaveno vstupem SS (*Slave select*) pro výběr komunikace, kdy komunikace se účastní vždy v daném okamžiku pouze dvě zařízení, tedy Master a jeden Slave. Sběrnice vyžaduje připojení vodiče pro hodinový signál SCK, pomocí něhož se provádí vyžadovaná synchronizace datového přenosu mezi dvěma komunikujícími zařízeními. Tento hodinový signál generuje pouze zařízení Master, pomocí pinu SS (*Slave select*) vybírá konkrétní podřízenou jednotku, zahajuje a ukončuje přenos dat. SS však není potřeba, pokud je na sběrnici připojen pouze jeden Slave. Softwarově (pokud nám to zařízení umožňuje) nebo hardwarově pak zajistíme automatické přivedení trvalého stavu log. 0 na vstupní pin SS, za účelem aktivace komunikační linky. Tím docílíme aktivaci komunikačních pinů Slave zařízení po celou dobu připojení. Zjednodušené blokové schéma SPI modulu MCU je zobrazeno na následujícím obr. 34.

Každý SPI modul je vybaven posuvným registrem, ke kterému je připojen generátor hodin (hodinový signál je poskytován oběma zařízením). Výstup posuvného registru zařízení Master je připojen na vstup posuvného registru zařízení Slave pomocí vodiče MOSI a vstup posuvného registru zařízení Master je připojen k výstupu posuvného registru zařízení Slave pomocí vodiče MISO. Oba posuvné registry jsou 8bitové.



Obr. 34: Zjednodušené *blokové schéma SPI rozhraní (podle* [1]*)*

Pokud tedy chce Master poslat bajt dat, umístí je do svého posuvného registru a po osmi pulzech hodinového signálu se celý obsah registru postupně přesune do registru Slave zařízení. Posuvný registr opouští nejprve nejméně významový bit (LSBit) až po ten nejvýznamnější (MSBit). Má-li dojít naopak k příjmu dat od Slave, pak je to právě Slave, který umístí bajt dat do svého posuvného registru a opět po osmi pulzech hodinového signálu dojde k přesunutí bajtu dat do registru Master. V blokovém schématu, SPI modulu, jsou navíc zobrazeny bloky Rx buffer a Tx buffer. Jedná se o vyrovnávací paměti, s kterými se můžeme setkat téměř u každého, dnes vyráběného, mikrokontroléru. Jsou prostředníkem mezi posuvným registrem a datovou sběrnicí během příjmu a vysílání dat. Jejich úkol je jednoduchý. Vysílaná data jsou z datové sběrnice nejprve uložena do Tx bufferu, pouze pokud je registr prázdný, data se vyzvednou a nahrají se do posuvného registru. Další data v Tx bufferu musí počkat na další vyprázdnění posuvného registru. Naopak, do Rx bufferu se ukládají přijatá data z posuvného registru a uchovávají se tak až do vyzvednutí. Přijatá data jsou tímto způsobem chráněna proti nežádoucí ztrátě. [1]

Princip komunikace [14]

Celý SPI protokol můžeme zjednodušeně popsat následovně. Chce-li Master data ze Slave číst, stáhne nejprve příslušný vodič SS připojeného k danému zařízení na aktivní úroveň, tedy log. 0, nastavená úroveň potrvá až do ukončení komunikace. Ostatní Slave zařízení neúčastnící se komunikace mají piny pro komunikaci ve vysoké impedanci, jsou tedy odpojena. To umožní aktivaci komunikačních pinů SPI komunikujícího Slave zařízení. Následně je od Masteru generován hodinový signál, s kterým jsou vysílaná data synchronizována po sběrnici. Master poté nejprve vyšle na MOSI příkaz pro čtení ze zařízení a následně byte adresy paměťového místa. Slave nakonec odpoví vysláním

dat na MISO. Chce-li Master data do Slave zapisovat, opět stáhne SS na aktivní úroveň až do ukončení komunikace. Master pak generuje hodinový signál, na MOSI vyšle příkaz pro zápis do Slave, adresu v paměti, kde chce zapisovat a nakonec odešle data, která se mají na dané adrese zapsat.

Nastavení komunikace

Používaným SCK vodičem se provádí synchronizace posunování obsahu posuvných registrů a vzorkování dat na datové lince. Formát komunikace závisí na polaritě hodinového signálu (CPOL), fází hodinového signálu (CPHA) a v neposlední řadě na formátu vysílaných nebo přijímaných dat. Tento nakonfigurovaný komunikační formát musí být stejný pro všechna zařízení připojených na společné SPI sběrnici. Jednotlivé nastavení komunikačního formátu se provádí příslušnými registry vybraného SPI modulu.

CPOL určuje nečinný klidový stav, kdy nedochází k přenosu dat na datové lince (MOSI, MISO) pomocí hodinového signálu na SCK. CPHA určuje okamžik čtení (vzorkování) dat. Během určené doby se data musí nacházet ve stabilním stavu. Celkem je tedy možné zvolit jednu ze čtyř kombinací a přizpůsobit si tak komunikaci podle vytvářené aplikace.

Tab. 1: *Vliv změny parametrů polarity a fáze na přenos dat*

CPOL = 0, **CPHA** = 0	Data se čtou s *náběžnou hranou* hodinového signálu, se *sestupnou hranou* dochází ke změně jednotlivých přenášených bitů.
CPOL = 0, **CPHA** = 1	Data se čtou se *sestupnou hranou* hodinového signálu, s *náběžnou hranou* dochází ke změně jednotlivých přenášených bitů.
CPOL = 1, **CPHA** = 0	Data se čtou se *sestupnou hranou* hodinového signálu, s *náběžnou hranou* dochází ke změně jednotlivých přenášených bitů.
CPOL = 1, **CPHA** = 1	Data se čtou s *náběžnou hranou* hodinového signálu, se *sestupnou hranou* dochází ke změně jednotlivých přenášených bitů.

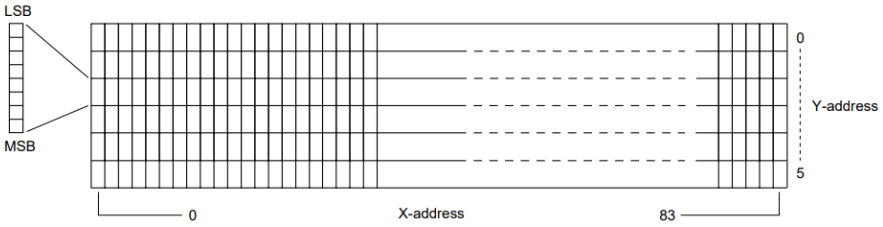
PDC8544 [5]

Jedná se o nízko příkonový řadič LCD displejů s maximálním rozlišením zobrazení 48x84.

Tab. 2: *Označení pinů použitého řadiče PDC8544*

$\boldsymbol{SDIN}$	Vstupní pin pro příjem dat (MOSI)
$\boldsymbol{SCLK}$	Vstupní pin pro hodinový signál z Masteru (max. 4 Mbit/s)
$\boldsymbol{D/\overline{C}}$	Určení přenosu, vysílání dat nebo příkazů
$\boldsymbol{\overline{SCE}}$	Chip enable (SS)
$\boldsymbol{\overline{RES}}$	Resetovací vstup (aktivní v log. 0)
$\boldsymbol{OSC}$	Oscilátor

Jak takový řadič pracuje, si můžeme ukázat na obr. 35. Zobrazení pixelů displeje je rozděleno do šesti hlavních bloků skládajících se z matice 8x84 pixelů. Prijatá data z SPI sběrnice jsou periodicky načítána z vnitřní paměti RAM řadiče. K určení umístění zobrazovaných znaků nám slouží ukazatel řádků X (0-84) a ukazatel sloupců Y (0-5), tyto ukazatelé se automaticky periodicky inkrementují. První přijatý byte dat je zobrazen na adrese X = 0, Y = 0, následně se inkrementuje obsah ukazatele X, další přijatý byte dat je tentokrát zobrazován na adrese X = 1, Y = 0. Zmíněným způsobem se zobrazí i ostatní data na bloku Y = 0 až dojde k přetečení (max. 84) nebo vynulování ukazatele X příslušným příkazem. Zároveň se inkrementuje ukazatel Y, všechna další data jsou zobrazována stejným způsobem, tedy opět se data zobrazují ve sloupcích X = 0 až X = 84, s tím rozdílem, že vše probíhá v dalším bloku Y = 1.



Obr. 35: *Princip zobrazování pomocí řadiče* [5]

Pomocí pinu $\boldsymbol{D/\overline{C}}$ určujeme v době 8. impulsu hodinového signálu, zda je vysílaný byte na ***SDIN*** příkazem (např. inicializace displeje) nebo zda jde o data pro zobrazení.

Po resetu displeje se provádí nejprve inicializace, kdy jsou nejprve zasílány příkazy adresy X a adresy Y paměti. Poté jsou postupně zasílána nulová data pro nulování vnitřní paměti. Tak se vynuluje paměť a připraví se pro zobrazování. Po inicializaci se zasílají data a příkazy do vnitřní paměti již jen pro zobrazování. Jak již bylo zmíněno, data jsou z paměti řadiče načítána a pomocí ukazatele řádků X a ukazatele sloupců Y postupně zobrazována. Inkrementace ukazatelů se provádí automaticky.

INSTRUCTION	D/C̄	COMMAND BYTE								DESCRIPTION
		DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(H = 0 or 1)										
NOP	0	0	0	0	0	0	0	0	0	no operation
Function set	0	0	0	1	0	0	PD	V	H	power down control; entry mode; extended instruction set control (H)
Write data	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	writes data to display RAM
(H = 0)										
Reserved	0	0	0	0	0	0	1	X	X	do not use
Display control	0	0	0	0	0	1	D	0	E	sets display configuration
Reserved	0	0	0	0	1	X	X	X	X	do not use
Set Y address of RAM	0	0	1	0	0	0	Y_2	Y_1	Y_0	sets Y-address of RAM; $0 \le Y \le 5$
Set X address of RAM	0	1	X_6	X_5	X_4	X_3	X_2	X_1	X_0	sets X-address part of RAM; $0 \le X \le 83$
(H = 1)										
Reserved	0	0	0	0	0	0	0	0	1	do not use
	0	0	0	0	0	0	0	1	X	do not use
Temperature control	0	0	0	0	0	0	1	TC_1	TC_0	set Temperature Coefficient (TC_x)
Reserved	0	0	0	0	0	1	X	X	X	do not use

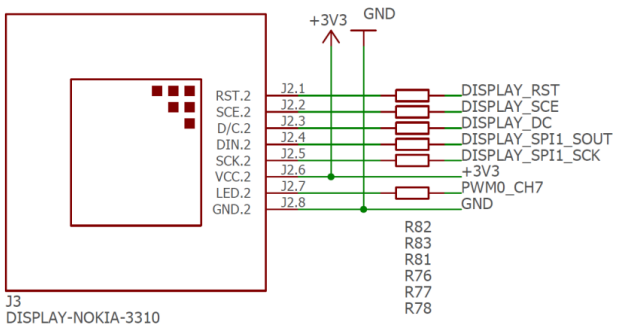
Během inicializace se zasílají příkazy 0x80h, 0x40h a nulová data 0x00h pro vynulování paměti.

Obr. 36: *Seznam instrukcí pro řízení zobrazení na displeji* [5]

2.5.3 Způsob vypracování a ověření výsledků

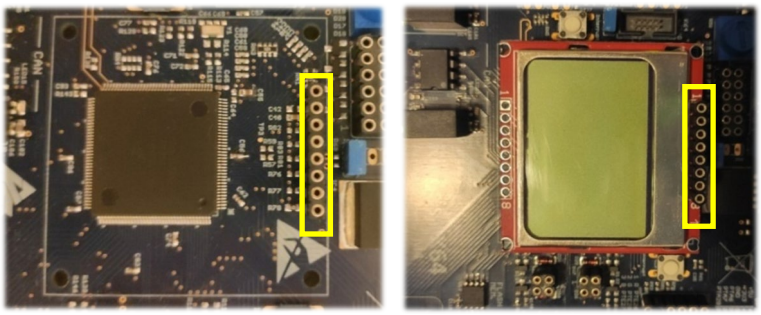
MCU je vybaven celkem třemi SPI moduly (SPI0, SPI1 a SPI2), které jsou připojeny ke společné vnitřní sběrnici *bus clock*. Generátor hodinového signálu je vybaven vnitřní děličkou frekvence s minimálním dělícím poměrem 2. Maximální dosažitelná rychlost je tedy *bus clock*/2. Každý modul umožňuje výběr jednoho z pracovních módů Master nebo Slave, podle požadavků programátora a vyvíjené aplikace. Moduly jsou vybaveny konfiguračními, stavovými a datovými registry, pro nastavení a řízení přenosu dat. Student nemusí tyto zmíněné registry složitě nastavovat, jelikož celé nastavení komunikačního protokolu je prováděno pomocí nástroje PE v grafickém okně.

Na laboratorní úloze si student vyzkouší především práci s SPI sběrnicí pro navázání průběhu komunikace s připojeným zařízením displeje s řadičem. K navázání komunikace se využívá modul SPI1. K fyzickému připojení displeje k desce s MCU slouží vyvedené konektory. Zobrazení si tak student správně otestuje a zhodnotí výsledky, ke kterým dospěl.



DISPLAY_RST	PTC9
DISPLAY_SCE	PTD4
DISPLAY_DC	PTC19
DISPLAY_SPI1_SOUT	PTD6
DISPLAY_SPI1_SCK	PTD5

Obr. 37: *Obvodové schéma zapojení displeje Nokia 3310 (vlevo) s popisem vyvedených pinů z mikrokontroleru (vpravo)*



Obr. 38: *Zapojení displeje Nokia 3310 s konektory na desce*

Vypracování

Pro vypracování aplikace slouží návod, jehož součástí je popis použití komponenty PDC8544, kterou student ke své práci využívá. Komponenta umožňuje nastavení a inicializaci displeje podle použitého řadiče, pomocí referenční komponenty SPIMaster_LDD se provádí nastavení parametrů

přenosu po SPI sběrnici, nastavení alternativní funkce zvolených pinů jako MOSI, MISO, hodinového synchronizačního signálu, nastavení šířky datového slova, směru vysílání a přijímání dat, volba polarity a fáze platnosti dat, nastavení frekvence hodinového signálu a mnoho dalšího. Jednoduché grafické nástroje jsou poskytnuty komponentou GDisplay, umožňuje nám vytvářet a libovolně umísťovat mj. jednoduché obrazce obdélníků, čtverců, kruhů po displeji. Kromě toho poskytuje bohaté nástroje pro zobrazení obrazců, využitím bajtového pole předdefinovaných prvků.

Používané funkce komponent

PDC8544 **(** PDC1:PDC8544**)**

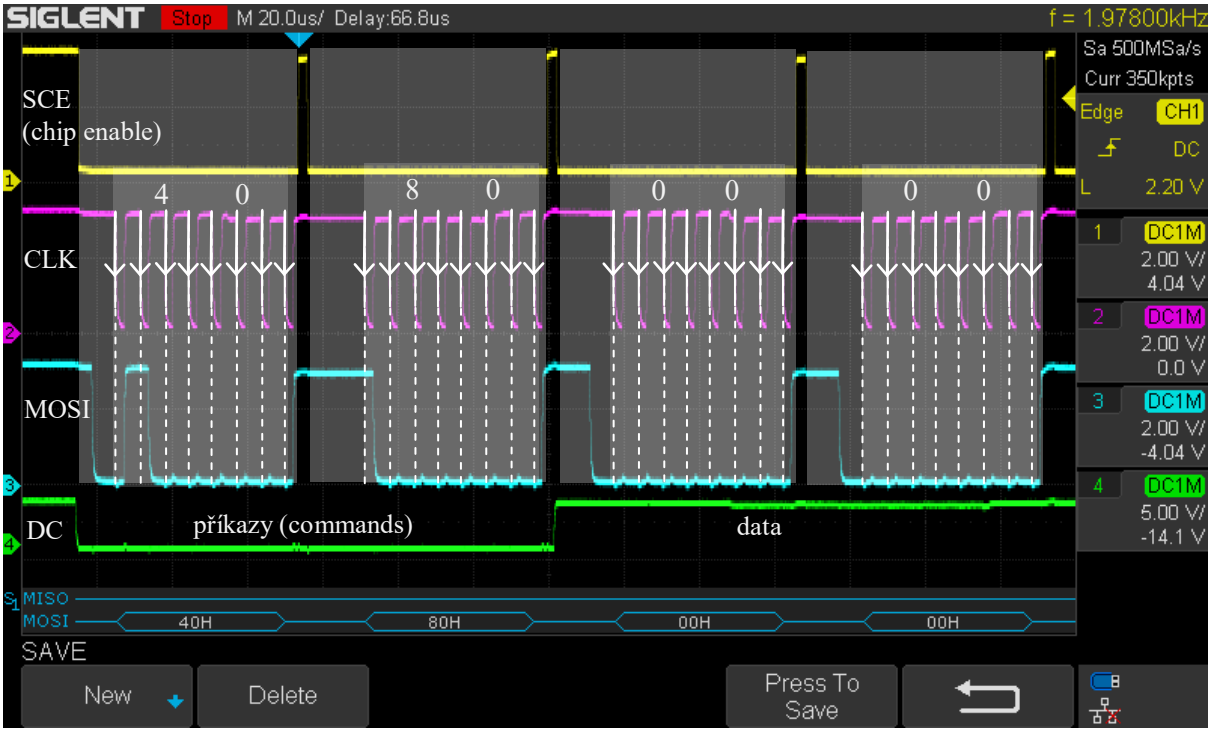
- *PDC1_SetPos* = pohyb kurzoru na zvolenou pozici na displeji v pixelech
- *PDC1_UpdateFull* = obnovení zobrazení celého displeje
- *PDC1_WriteLineStr* = výpis textu na displeji

GDisplay **(** GDisp1:GDisplay **)**

- *GDisp1_DrawFilledBox* = zobrazení vyplněného boxu nastavených rozměrů a umístění
- *GDisp1_DrawMonoBitmap* = vykreslení obrázku pomocí bajtového pole s umístěním pozice

Ověření výsledků

Kromě displeje, má student k dispozici osciloskop, který je vybaven funkcí dekódování SPI komunikace. To mu pomůže ověřit si znalosti získaných z přečtení teoretického rozboru a postupem vypracování laboratorní úlohy dle návodu. Po resetu zařízení je možné se přesvědčit, jak probíhá nulování vnitřní paměti řadiče v rámci inicializace před zobrazováním. Student si tak potvrdí informace, jak se rozlišují příkazy od dat, jejž se zapisují do vnitřní paměti, jaký vliv má nastavení polarity a fáze synchronizačního hodinového signálu na celkový formát komunikačního protokolu. V závěru se tak student vyjadřuje k dosaženým výsledkům z přiložených snímků obrazovky z osciloskopu, viz následující obrázek. Na obr. 39 je zobrazená dekódovaná komunikace s vyznačením časových okamžiků platnosti přenášených dat synchronizujícího se podle nastaveného hodinového signálu.



Obr. 39: *Přiblížení dekódované komunikace po sběrnici SPI mezi MCU a řadičem displeje*

3 Vypracované protokoly

3.1 Požadavky na vytvoření protokolů

Vzorové vypracované protokoly jsou přiloženy v příloze bakalářské práce. Každý protokol by měl v první řadě obsahovat zadání laboratorní úlohy. Dále by měl student zahrnout do svého vypracování vlastní teoretický rozbor, přiložený funkční programový kód, snímky z obrazovky osciloskopu zachycující funkčnost dané naprogramované aplikace, popřípadě také dodat pořízené snímky, pokud to daná laboratorní úloha umožňuje. Student by měl dále správně popsat funkčnost napsaného programu a doložit tak, že se v dané problematice orientuje. Příklad, jak má přiložený program správně vypadat je zobrazeno na obr. 40.

C:\Users\PC\Desktop\BP_CMT2\Lab1_Aplikace.c 1

```
1  #include "Aplikace.h"
2
3  void setup(void) {
4      TI1_EnableEvent();          /* povoleni preruseni casovacem */
5  }
6  /* generovani casoveho zpozdeni pomoci CPU (funkce waitms), generovani
     obdelniku T = 200 ms */
7  void loop(void) {
8      LED1_On();                  /* rozsviceni led na PTB2  */
9      WAIT1_Waitms(100);          /* cas. zpozdeni 100ms */
10     LED1_Off();                 /* zhasnuti led na PTB2  */
11     LED2_On();                  /* rozsviceni led na PTB3  */
12     WAIT1_Waitms(100);          /* cas. zpozdeni 100ms */
13     LED2_Off();                 /* zhasnuti led na PTB3  */
14 }
15 /* generovani cas. zpozdeni pomoci preruseni od casovace (2 ms), generovani
     obdelniku T = 4s */
16 void isr_pit0(void) {
17     LED3_Neg();                 /* negace led na PTB4  */
18 }
19
```

Obr. 40: *Aplikace s okomentováním funkce a práce napsaného kódu*

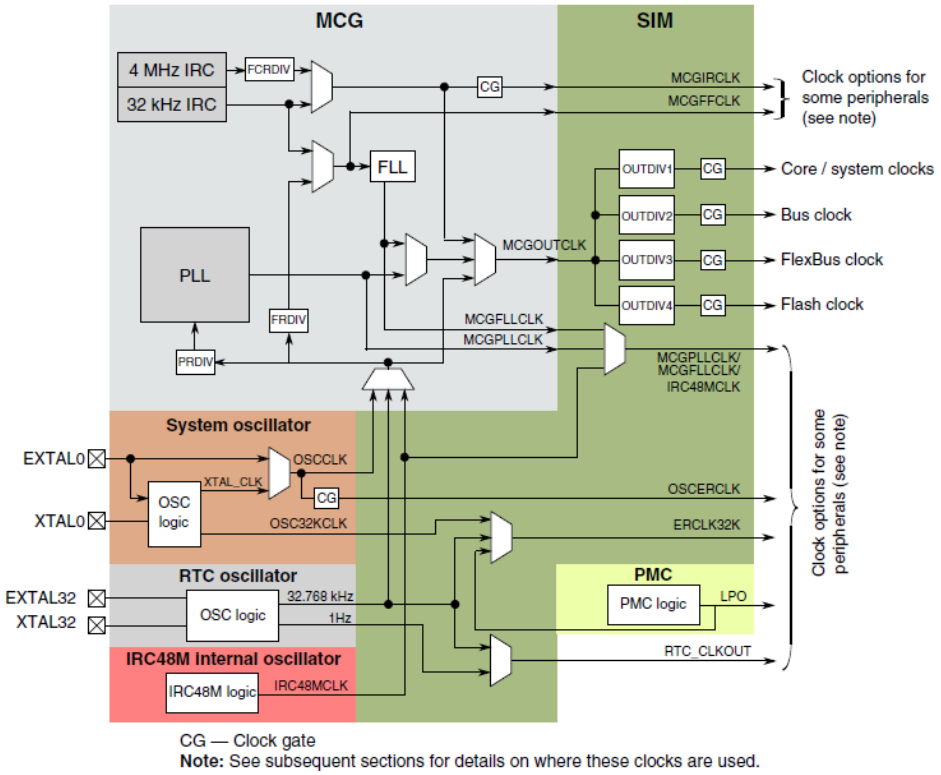
V závěru je úkolem zhodnotit výsledky, kterých student dosáhl. Student je porovnává s teoretickými předpoklady, vyjadřuje se k dané problematice, vhodně odůvodňuje, co se potvrdilo, čeho se naopak nepovedlo dosáhnout. Cílem je klasifikovat chyby, kterých se sám během řešení dopustil a přijít s vlastním řešením k odstranění problémů.

4 Podpůrné materiály

Tato kapitola slouží pro zobrazení konfigurace nejčastěji používaných komponent v laboratorních úlohách pomocí nástroje PE. Nejdůležitější řádky nastavení jsou popsány tak, aby bylo možné zjistit, co bylo úkolem nastavení určité jednotlivé konfigurace.

4.1 Nastavení časování

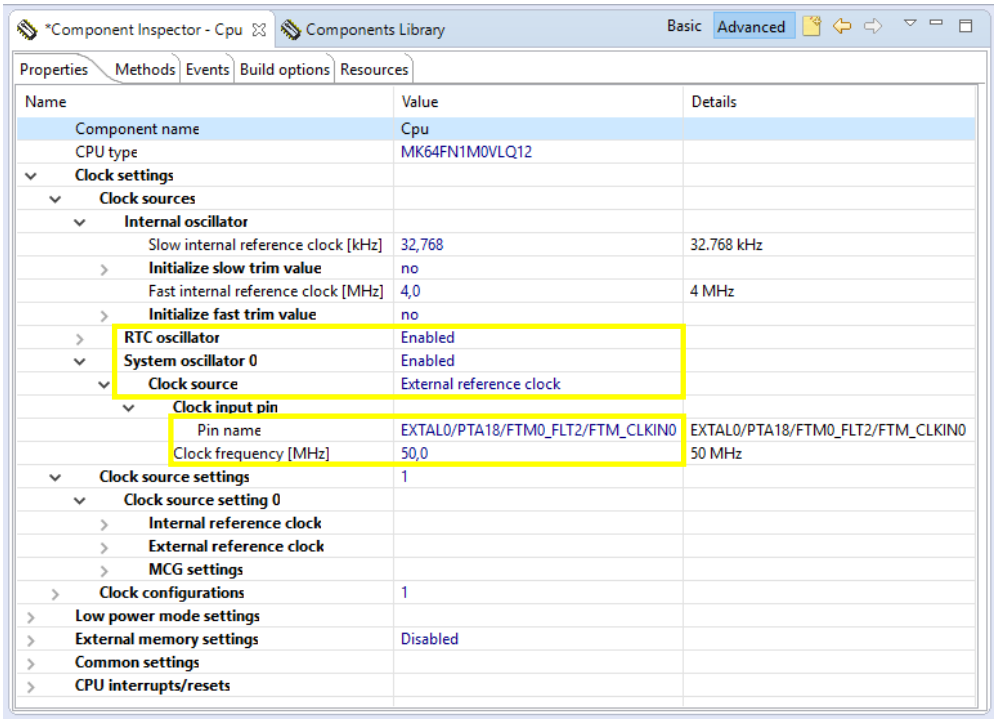
Jednou z nejdůležitějších a prvotních nastavení každého mikrokontroléru patří nastavení časování, tedy řízení výběru hodinového signálu pro všechny obvody a periférie. Diagram časovacího obvodu zobrazující jednotlivé zdroje hodinových impulsů pro nastavení hodin všech periferií, včetně hodin samotného procesoru mikrokontroléru **MK64FN1M0VLQ12** zachycuje následující obrázek.



Obr. 41: *Blokové schéma časovacího obvodu mikrokontroléru* [6]

Mikrokontrolér disponuje několika zdroji hodinových signálů. Je tedy možné vybírat z mnoha nastavení. Správným nastavením můžeme dosáhnout maximální frekvence *Core clock* a *System clock* 120 MHz, 60MHz pro *Bus clock a External bus clock*, 24 MHz *Flash clock*. Kromě nastavení maximální frekvence máme k dispozici VLPR mode, tedy nastavení velmi nízké spotřeby. *Core clock* a *System clock*, *Bus clock*, *External bus clock* pak mohou pracovat s frekvencí méně než 4 MHz a *Flash clock* s frekvencí pod 1 MHz. Více informací se můžete dozvědět z referenčního manuálu. Pro dosažení nejvyššího výpočetního výkonu, tedy 120 MHz pro CPU, je nutné postupovat dle následujícího návodu.

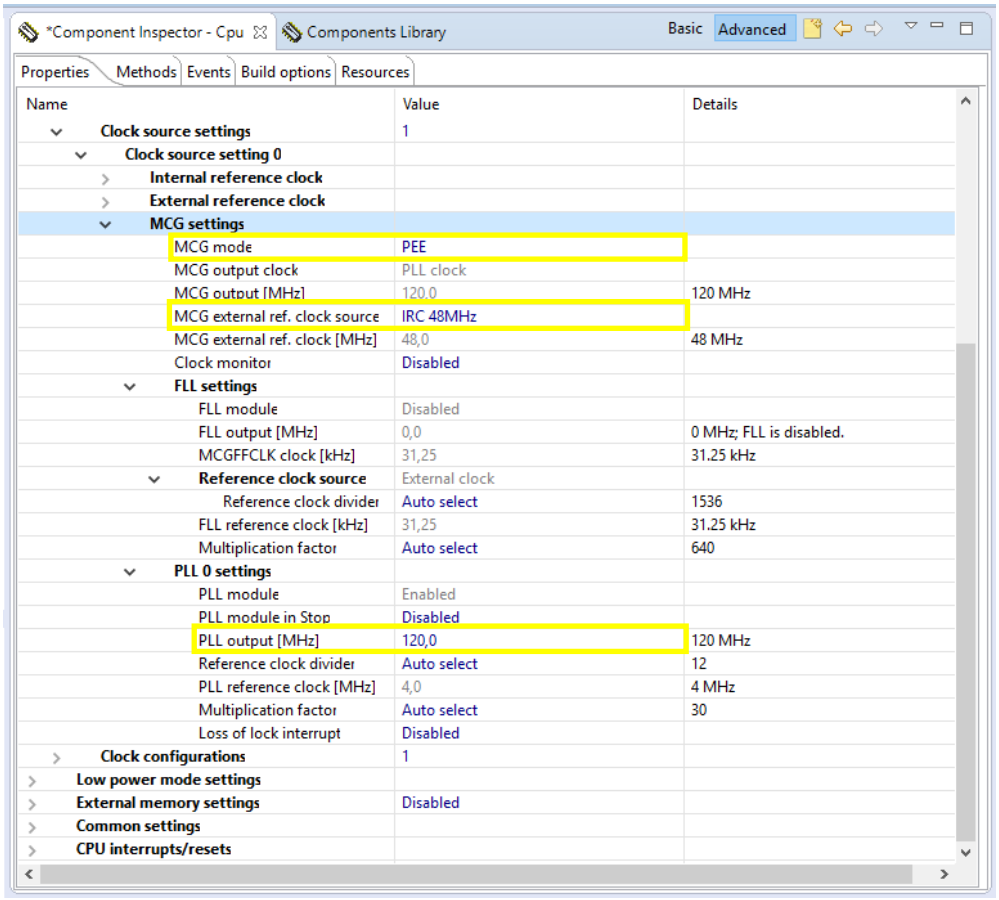
Jedním z možných zdrojů frekvence je RTC oscilátor tvořený 32 kHz krystalem, který je připojený na pinech XTAL32 a EXTAL32. Tento zdroj má oddělené napájení. Systémový oscilátor je tvořen 25 MHz krystalem, který je pomocí obvodu PHY upraven na hodnotu 50 MHz. Od této frekvence je již odvozena vstupní frekvence pro vnitřní obvod oscilátoru (vstupní piny XTAL0 a EXTAL0).



Obr. 42: *Nastavení časování pro dosažení frekvence 120 MHz*

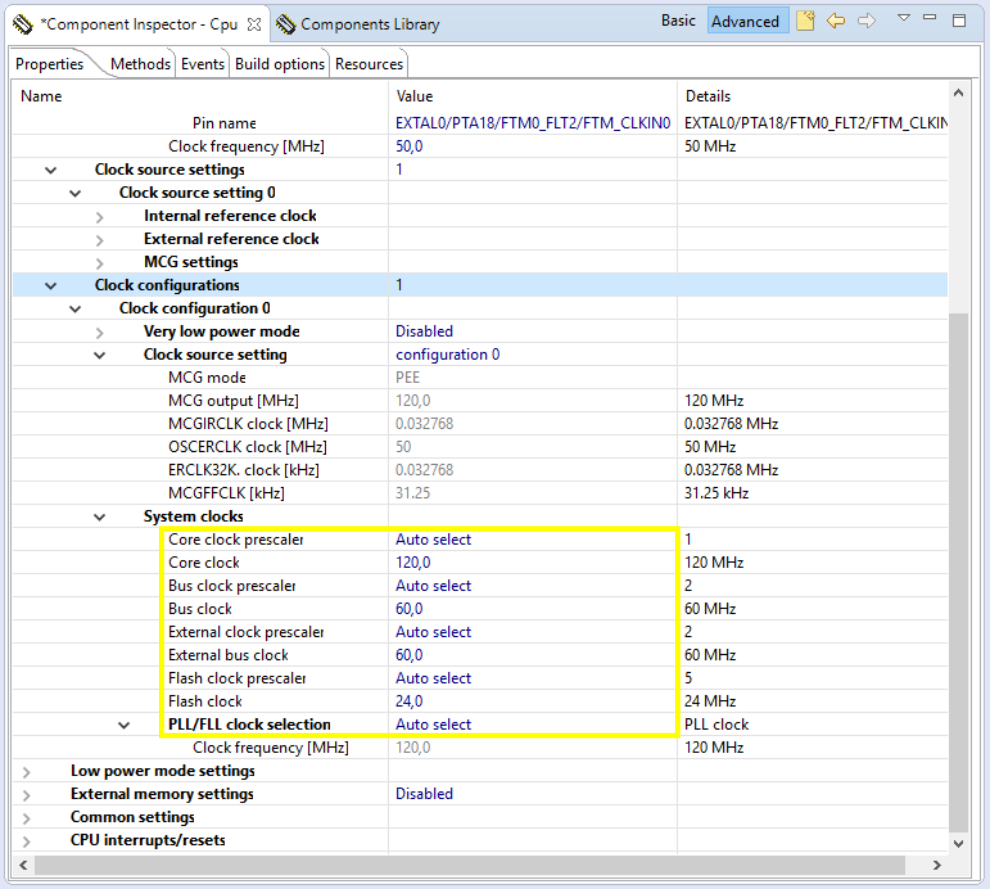
Uvnitř mikrokontroléru se také nachází 48 MHz vnitřní RC oscilátor IRC, který budeme projektech využívat jako referenční hodinový signál, od jehož frekvence budou odvozeny hodinové signály pro jednotlivé části mikrokontroléru. Pro vytvoření námi požadované frekvence 120 MHz máme k dispozici obvod PLL (*Phase-locked loop*) a FLL (*Frequency-locked loop*).

Blok časování je vybaven obvodem MCG (*Multi-clock generator*), který nám slouží nejen pro výběr vnitřního zdroje hodinového signálu 4MHz IRC nebo 32kHz IRC, ale také úpravu frekvence, ze zvoleného zdroje referenčního signálu, kterou získáváme na výstupu MCG *output*, pomocí vnitřních multiplexerů a děliček frekvence. MCG nám umožňuje výběr z 8 módů **(**MCG *mode***)**. Výběrem módu PEE (*PLL Engaged External*) je výstupní frekvence z MCG (*MCG output*) odvozena z PLL, který je kontrolovaný z externího referenčního hodinového signálu. Výběr externího referenčního hodinového signálu je řízen pomocí MCG *ref. clock source*. Máme možnost zvolit si jeden z 3 externích zdrojů hodinového signálu. V našem případě volíme vnitřní obvod oscilátoru IRC48M. Zvolením výstupní frekvence PLL obvodu (PLL *output*) nám *Processor Expert* automaticky vypočítá nutný dělící a násobící poměr pro dosažení námi zvolených 120 MHz.



Obr. 43: *Nastavení bloku MCG pro časování*

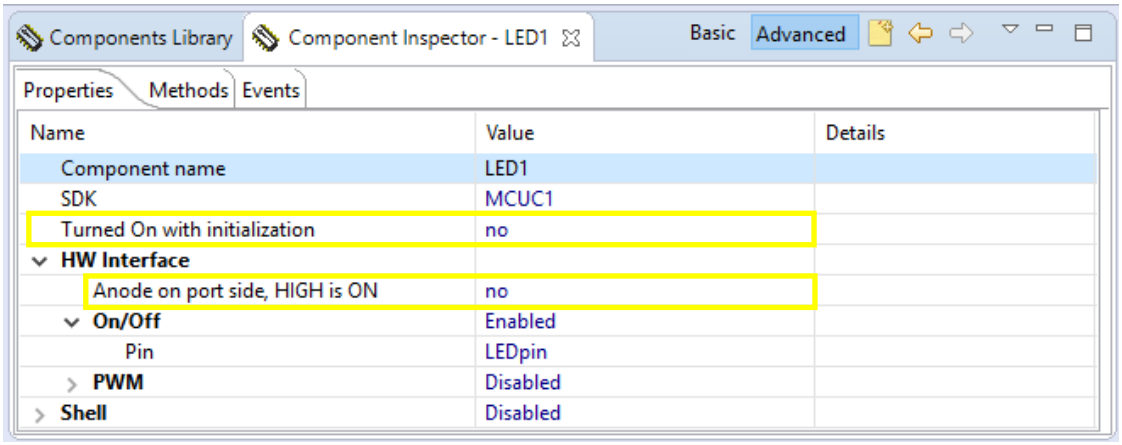
Následně je potřeba nastavit jednotlivé frekvence pro procesor (*Core clocks*), vnitřní sběrnici (*Bus clock*), externí sběrnici (*FlexBus clock*) a také pro paměť programu (*Flash clock*). K tomuto účelu je časovací obvod vybaven modulem SIM *(System Integration Module*), který poskytuje velké množství konfiguračních registrů pro výběr hodinového signálu a řízení úpravy frekvence pomocí děliček a multiplexerů. Po zvolení hodnot jednotlivých frekvencí dílčích částí nám Processor Expert sám vypočítá potřebný dělící poměr děliček frekvence pro dosažení těchto frekvencí.



Obr. 44: *Nastavení časování systému*

4.2 Nastavení komponenty *LED*

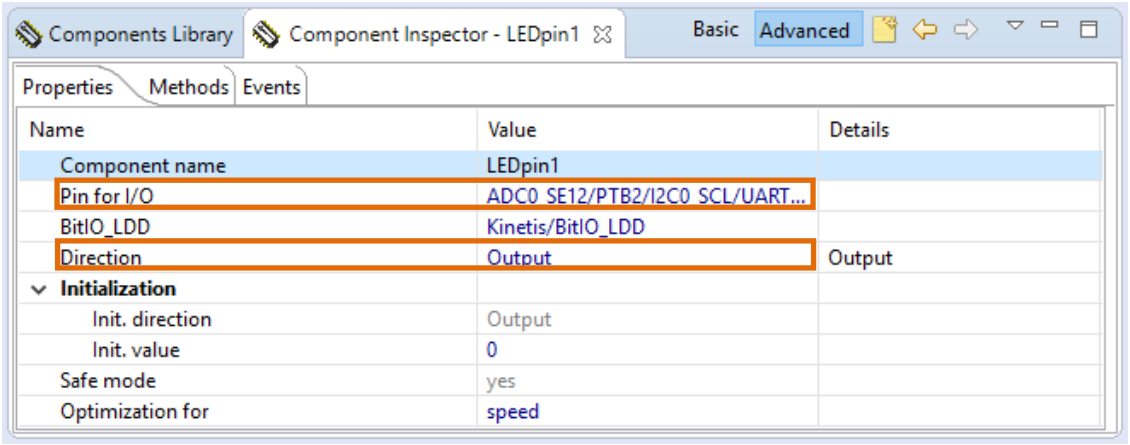
Komponenta slouží pro jednoduché a rychlé nastavení pinu pro ovládání led. Její součástí je podřízená komponenta BitIO. U nastavení nadřazeného bloku komponenty postupujeme následujícím způsobem.



Obr. 45: *Nastavení komponenty LED1*

Turned On with inicialization (no) … výběr počátečního stavu led po provedení inicializace, (zhasnuto)
Anode on port side (no) … zvolená led je svou katodou připojena k výstupnímu pinu, pak tedy svítí přivedením log. 0

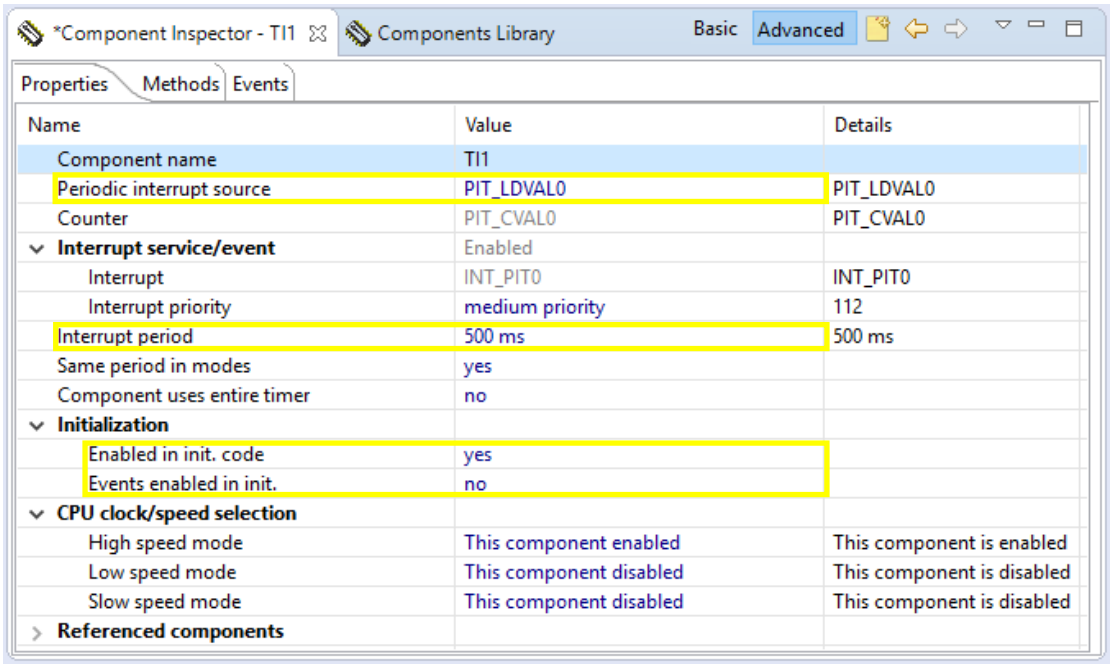
Pomocí podřízené komponenty BitIO inicializujeme zvolený pin portu B, na kterém je daná led připojena, následovně:



Obr. 46: *Nastavení podružné komponenty BitIO*

Pin for I/O (PTB2) … volba pinu 2 portu B, na kterém je led připojena
Direction (Output) … volba funkce pinu (výstupní)

4.3 Nastavení komponenty časovače *TimerInt*



Obr. 47: *Nastavení komponenty TimerInt*

Periodic interrupt source (PIT_LDVAL0) … výběr zdroje přerušení (časovače)
Interrupt period (500 ms) … nastavení času přerušení
Enabled in init. code (yes) … povolení automatického spuštění časovače během inicializace
Events enabled in init. (no) … zakázání automatického povolení přerušení časovače (povolení nebo zákaz budeme provádět programově pomocí funkcí *EnableEvent/DisableEvent*, které je před použitím potřeba povolit)

4.4 Nastavení komponenty *AsynchroSerial*

Component Inspector - AS1 | Components Library | Basic | Advanced

Properties | Methods | Events

Name	Value	Details
Component name	AS1	
Channel	UART4	UART4
Interrupt service/event	Enabled	
Interrupt RxD	INT_UART4_RX_TX	INT_UART4_RX_TX
Interrupt RxD priority	medium priority	112
Interrupt TxD	INT_UART4_RX_TX	INT_UART4_RX_TX
Interrupt TxD priority	medium priority	112
Interrupt Error	INT_UART4_ERR	INT_UART4_ERR
Interrupt Error priority	medium priority	112
Input buffer size	64 D	
Output buffer size	256 D	
Handshake		
CTS	Disabled	
RTS	Disabled	
Settings		
Parity	none	none
Width	8 bits	8 bits
Stop bit	1	1
Receiver	Enabled	
RxD	ADC0_SE18/PTE25/UART4_RX/I2C0_...	
Transmitter	Enabled	
TxD	ADC0_SE17/PTE24/UART4_TX/I2C0_...	
Baud rate	115200 baud	115163.148 baud
Break signal	Disabled	
Wakeup condition	Idle line wakeup	
Transmitter output	Not inverted	
Receiver input	Not inverted	
Stop in wait mode	no	
Idle line mode	starts after start bit	
Break generation length	Short	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	
CPU clock/speed selection		
High speed mode	This component enabled	This component is enabled
Low speed mode	This component disabled	This component is disabled
Slow speed mode	This component disabled	This component is disabled
Referenced components		
Serial_LDD	Kinetis/Serial_LDD	

Obr. 48: *Nastavení komponenty AsynchroSerial*

Channel (UART4) **…** zvolený komunikační modul, *UART4* je vyvedený ke komunikačnímu rozhraní *OpenSDA*, jenž nám bude zprostředkovávat samotnou komunikaci mezi *PC* a *MCU*

Interrupt service/event (Enabled) … povolení přerušení při komunikaci se sériovým kanálem

Input buffer size (64) … velikost vstupního bufferu pro ukládání přijatých dat v bytech

Output buffer size (256) … velikost výstupního bufferu sloužícího pro ukládání vysílaných dat v bytech k následnému přenosu

Receiver (UART4_RX) … nastavení vstupního pinu pro příjem dat modulu *UART4*

Transmitter (UART4_TX) … nastavení výstupního pinu pro vysílání dat modulu *UART4*

Baud rate (115200 baud) … volba přenosové rychlosti komunikace (musí být stejná pro obě zařízení účastnící se komunikace po sériové lince)

Inicialization (yes/yes) … nastavení automatické inicializace, v opačném případě se inicializace musí provádět pomocí příslušných funkcí komponenty před začátkem zahájení komunikace (*Enable, Enable Event*)

Závěr

Povedlo se vypracovat poměrně obsáhlé návody a programové kódy všech laboratorních úloh, které slouží jako podpůrný materiál pro vyučujícího, programy jsou navíc doplněny o poznámky. Dále bylo zhotoveno vzorové vypracování protokolů pro názornou ukázku správného vypracování. Povedlo se tedy dát jak studentovi, tak vyučujícímu kompletní materiál a podklady pro výuku a studium základních, nejfrekventovaněji používaných modulů mikropočítače, kterými jsou GPIO, čítače a časovače (tak také speciální FTM moduly), komunikační periférie UART, I2C či SPI. Student je také postupně, již před započetím práce, proveden možnostmi vypracování požadované aplikace a je uveden do dané problematiky.

Při řešení aplikace pro laboratorní úlohu č. 4 se vyskytly těžkosti s komunikací s teplotním senzorem přes I2C sběrnici, proto bylo potřeba programový kód správně doplnit o startovací sekvenci pro navázání komunikace.

Jelikož se jedná o práci zaměřenou na vytvoření výukových materiálů, nabízí se zde poměrně velký prostor pro budoucí vylepšení úloh či doplnění práce na další laboratorní úloze, pokud by to tedy časový plán ve cvičeních umožňoval.

Jako možné vylepšení vidím v doplnění laboratorní úlohy pro rozšíření vědomostí studenta v oblasti využití čítačů a časovačů v režimu *intput capture*. Jednalo by se o úlohu, která by zahrnovala použití jednoduchého modulu ultrazvukového senzoru za účelem vyhodnocení vzdálenosti měřeného objektu či pouze detekce překážky. Během zpracovávání aplikace by se student seznámil s bližší prací s ukazateli či s velmi často používanými strukturami, které usnadňují a zpřehledňují práci během vytváření programového kódu. Tuto práci se strukturami a ukazateli jsem již naznačil v bonusovém zadání laboratorní úlohy č. 5, kde se provádí konverze obrázku nakresleného v jednoduchém grafickém editoru s rastrovou grafikou na formát bajtového pole s jeho následným zpracováním a zobrazením na displeji.

Jelikož se jedná o předmět pro studenty, kteří se velmi často setkají nebo již setkali v průběhu studia, teoreticky či již prakticky, s prací a s principy vyhodnocování otáček kol či motoru, bylo by zajímavé a velmi přínosné přidat speciálně vytvořenou laboratorní úlohu pro vyhodnocování otáček hřídele prostřednictvím zpracování signálů z rotačního enkodéru, který slouží jako záporná zpětná vazba pro řízení. Využil by se také větší potenciál vývojové desky s MCU a jeho FlexTimer modulu, který má speciální funkce věnované aplikacím pro řízení motorů. Na PWM konektoru jsou tedy vyvedeny přímo vstupní piny pro vyhodnocení výstupního signálu právě z rotačního enkodéru sloužícího pro přesné vyhodnocení polohy natočení hřídele motoru nebo měření úhlové rychlosti otáčení. Laboratorní úloha by vycházela ze získaných poznatků z předchozích laboratorních úloh. Zpracované informace by se zobrazovaly například na displeji nebo také na komunikační terminál PC.

Věřím, že se má práce stane dobrým nástrojem ve výukových hodinách a stane se tak naplněním mého cíle této práce a to zpříjemnit výuku, usnadnit práci v množství dostupných zpracovaných informací a posílit tak ve studentovi nadšení v mikropočítačovém odvětví elektroniky, které je dnes na vzestupu a zaslouží si tedy značnou pozornost každého studenta jakéhokoliv oboru zaměřeného na elektroniku.

Literatura

[1] MAZIDI, Muhammad Ali, Sarmad NAIMI, Sepehr NAIMI a Shujen CHEN. *Freescale ARM Cortex-M embedded programming: using C language*. B.m.: Microdigitaled, 2016. ISBN 978-0-9979259-8-2.

[2] *Referenční manuál MK64FN1M0VLQ12* [online]. [vid. 2020-03-28]. Dostupné z: https://www.mouser.com/datasheet/2/813/K64P144M120SF5RM-1074828.pdf

[3] *OpenSDA* [online]. [vid. 2020-03-29]. Dostupné z: https://www.nxp.com/docs/en/user-guide/OPENSDAUG.pdf

[4] *I2C-bus specification and user manual* [online]. [vid. 2020-03-29]. Dostupné z: https://www.nxp.com/docs/en/user-guide/UM10204.pdf

[5] *PCD8544 PhilipsSemiconductors* [online]. [vid. 2020-03-29]. Dostupné z: https://cdn.datasheetspdf.com/pdf-down/P/C/D/PCD8544_PhilipsSemiconductors.pdf

[6] *Kinetis K64: 120MHz Cortex-M4F up to 1MB Flash 100-144pin* [online]. [vid. 2020-03-30]. Dostupné z: https://www.nxp.com/webapp/Download?colCode=K64P144M120SF5RM

[7] STYGER, Erich. Overview: Processor Expert. *MCU on Eclipse* [online]. 18. říjen 2015 [vid. 2020-03-29]. Dostupné z: https://mcuoneclipse.com/2015/10/18/overview-processor-expert/

[8] STYGER, Erich. Mother of Components: Processor Expert with NXP Kinetis SDK V2.0 Projects. *MCU on Eclipse* [online]. 15. květen 2016 [vid. 2020-03-29]. Dostupné z: https://mcuoneclipse.com/2016/05/15/mother-of-components-processor-expert-with-nxp-kinetis-sdk-v2-0-projects/

[9] *Periodic Interrupt Timer* [online]. [vid. 2020-03-29]. Dostupné z: https://www.nxp.com/docs/en/supporting-information/Periodic-Interrupt-Timer-Training.pdf

[10] PRAUZEK, Michal. *ČÍSLICOVÁ A MIKROPROCESOROVÁ TECHNIKA*. Ostrava: VŠB-TU, 2013. ISBN xxxxxxxxx.

[11] *FlexTimer Module* [online]. [vid. 2020-03-29]. Dostupné z: https://www.nxp.com/docs/en/application-note/AN5142.pdf

[12] *I2C manual* [online]. [vid. 2020-03-29]. Dostupné z: https://www.nxp.com/docs/en/application-note/AN10216.pdf

[13] *LM75A Digital temperature sensor* [online]. [vid. 2020-03-31]. Dostupné z: https://www.nxp.com/docs/en/data-sheet/LM75A.pdf

[14] *Sériová rozhraní* [online]. [vid. 2020-03-29]. Dostupné z: http://home.zcu.cz/~dudacek/NMS/Seriova_rozhrani.pdf

Přílohy